CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Stabilized Structured Dantzig-Wolfe Decomposition Method

Antonio Frangioni
Bernard Gendron

Janvier 2010

CIRRELT-2010-02

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

A Stabilized Structured Dantzig-Wolfe Decomposition Method

Antonio Frangioni[1], Bernard Gendron[2,*]

[1] Dipartimento di Informatica, Università di Pisa, Polo Universitario della Specia, Via dei Colli 90, 19121 La Specia, Italy

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and Department of Computer Science and Operations Research, Université de Montréal, C.P. 6128, succursale Centre-Ville, Montréal, Canada H3C 3J7

Abstract. We present an algorithmic scheme, which we call the Structured Dantzig-Wolfe decomposition method, that can be used for solving large-scale structured Linear Programs (LPs). The required structure of the LPs is the same as in the original Dantzig-Wolfe approach, that is, a polyhedron over which optimization is "easy" plus a set of "complicating" constraints. Under the hypothesis that an alternative model for the "easy" polyhedron and a simple map between the formulations are available, we propose a finite procedure which solves the original LP. The original Dantzig-Wolfe decomposition method is easily seen to be a special case of the new procedure corresponding to the standard choice for the model of the "easy" polyhedron where there is a variable for each of its extreme points and rays. Furthermore, some techniques developed for the standard DW decomposition can be easily adapted to the new approach, giving rise to Stabilized Structured Dantzig-Wolfe decomposition methods that can be more efficient than their non-stabilized brethren. We show the usefulness of our new approach with an application to the multicommodity capacitated network design problem, exploiting some recent results which characterize the polyhedral properties of the multiple choice formulation of the problem.

Keywords. Dantzig-Wolfe decomposition method, structured Linear Program (LPs), multicommodity capacitated network design problem, reformulation, stabilization.

Acknowledgements. We are grateful to Serge Bisaillon for his help with implementing and testing the algorithms. We also gratefully acknowledge financial support for this project provided by Natural Sciences and Engineering Council of Canada (NSERC), and by MIUR (Italy) under the PRIN projects "Optimization, simulation and complexity in telecommunication network design and management" and "Models and algorithms for robust network optimization".

---

* Corresponding author: Bernard.Gendron@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2010

1 Introduction

The celebrated Dantzig-Wolfe (DW) decomposition method [12], inspired by an algorithm due to Ford and Fulkerson [13] for optimal multicommodity flow computations, can be considered one of the most far-reaching results of mathematical programming. Although in its original development it was strongly linked with the inner workings of the simplex algorithm, its current form is very elegant and easy to understand and implement. It allows to efficiently deal with problems with a very general and common structure, that is

$$(P) \qquad \min_x \{\, cx : \; Ax = b \,,\; x \in X \,\}$$

where the set X is "easy", i.e., linear optimization over X is "significantly easier" than (P). In other words, (P) would be an "easy" problem if the *complicating* constraints $Ax = b$ could be removed, i.e., the constraints $Ax = b$ break down the structure of X. One typical example is that of a *separable* X (see Section 2.1.2), i.e., the one where (P) would decompose into a number of smaller independent subproblems if the *linking* constraints $Ax = b$ could be removed. This is for instance the case of multicommodity flow problems, such as those who originally inspired the DW approach [13] and our application described in Section 4. The DW method is strongly tied with Lagrangian algorithms; in fact, it can be shown (see, for instance, [17, 25]) to be equivalent to Kelley's cutting-plane method [24] applied to the Lagrangian Dual of the original problem. Hence, a large number of algorithms can be considered to be variants of the DW approach, as discussed in [16, 17, 25].

In this paper, we extend the DW approach, building a broader algorithmic paradigm that encompasses and generalizes it. The essential observation that leads to this new paradigm, broadly discussed in the following, is that the DW algorithm is based on a *reformulation* approach. In fact, the idea behind the DW algorithm is that of substituting a formulation of the "easy" polyhedron X in the original variables with a reformulation in a different space of variables, namely the convex (and conical) multipliers that allow to express any point of X as a convex combination of its extreme points (plus a conical combination of its extreme rays). Optimization over X can then be seen as a *variable generation procedure*, in that it allows to either add to the partial formulation one new relevant variable, or to establish that the current partial formulation already provides the optimal solution.

Under appropriate conditions, it is possible to mirror the DW approach using any other model of X. In order for the approach to be of interest, the model must be "large", i.e., the number of constraints and variables must be such that it is not convenient to insert the whole model in one single LP and solve the whole problem at once. However, as shown in the actual application to the multicommodity capacitated network design problem (see Section 4), the model can be "significantly smaller" than that of the classical DW approach, e.g. by having "only" pseudo-polynomially many constraints and variables as opposed to exponentially many variables (and maybe few constraints, except the simple ones). As our computational experience shows, this can have an impact on the relative efficiency of the corresponding decomposition procedures. Because our approach further exploits structural information on the problem to define a proper model of X, we call it the *Structured Dantzig-Wolfe (SDW) decomposition approach.*

One interesting feature of the SDW approach is that it does not change most the requirements of the original DW algorithm. In other words, provided that a proper map can be established between a solution in the original space of variables and the one used for reformulating X, the subproblem to be solved at each iteration remains the same; only the master problem changes. Hence, if an implementation of the DW approach is available for one problem for which an alternative model of

X is known, then implementing the SDW approach for that problem only requires relatively minor changes to the existing code. Furthermore, we will show that the SDW approach can be *stabilized* exactly in the same way as the standard DW. This gives rise to *Stabilized Structured Dantzig-Wolfe* (S^2DW) algorithms, that can be analyzed by the general theory developed in [16], which provides a comprehensive convergence analysis for a large choice of *stabilizing terms* and several different algorithmic options.

The structure of the paper is as follows. In Section 2, the original Dantzig-Wolfe decomposition method is first reviewed, then the Structured Dantzig-Wolfe decomposition method is presented and discussed. Section 3 is devoted to describing how the SDW approach can be stabilized, providing the appropriate discussion and convergence results. Section 4 presents an application of the SDW approach to the Multicommodity Capacitated Network Design problem (MCND); in particular, the results of extensive computational experiments comparing SDW and S^2DW with other alternatives are presented and discussed. Finally, conclusions are drawn in Section 5.

2 The Structured Dantzig-Wolfe Decomposition Method

The DW approach can be used to solve the *convexified relaxation* of (P)

$$\min_x \{ cx : Ax = b\ ,\ x \in conv(X) \} \tag{1}$$

where $conv(\cdot)$ denotes the convex hull. If X is convex then the DW approach directly solves (P), otherwise the (repeated) solution of (1) can be instrumental for solving (P), e.g. within an enumerative (Branch&Bound) approach. It is well-known [17, 25] that solving (1) is equivalent to forming the *Lagrangian relaxation* of (P) with respect to the complicating constraints $Ax = b$ and one generic vector π of *Lagrangian multipliers*

$$(P_\pi) \qquad f(\pi) = \min_x \{ cx + \pi(b - Ax) \ :\ x \in X \}$$

and solving the corresponding *Lagrangian dual*

$$\max_\pi \{ v(P_\pi) \ :\ \pi \in \mathbb{R}^m \} \tag{2}$$

where $v(\cdot)$ denotes the (possibly infinite) optimal objective function value of an optimization problem and m denotes the number of complicating constraints $Ax = b$. The *Lagrangian function* $f(\pi) = v(P_\pi)$ is well-known to be concave; thus, (2) is an "easy" problem. As rapidly recalled below, solving (2) by means of Kelley's cutting-plane method [24] is equivalent to solving (1) by the DW approach. This provides both an optimal solution x^* of (1) and an optimal solution π^* of (2).

2.1 Dantzig-Wolfe Decomposition

To simplify the exposition, we temporarily assume that X is a *compact* set; later on in this section, we recall how this assumption can be dropped at the only cost of slightly complicating the notation. We also assume that X is nonempty, otherwise (P) is not feasible (this is discovered the first time (P_π) is solved, whatever the vector π).

As anticipated in the introduction, the fundamental idea under the Dantzig-Wolfe decomposition algorithm is that of *reformulation*. The simple observation is that (1) can be rewritten as the large-scale (possibly even semi-infinite) Linear Program

$$\min_\theta \{ c\left(\textstyle\sum_{x \in X} x\theta_x\right) \ :\ A\left(\textstyle\sum_{x \in X} x\theta_x\right) = b\ ,\ \theta \in \Theta \} \tag{3}$$

where $\Theta = \{ \theta \geq 0 : \sum_{x \in X} \theta_x = 1 \}$ is the unitary simplex of proper dimension. In most cases (3) is an ordinary LP because only a finite set of "relevant" points of $x \in X$ need to be associated a convex multiplier θ_x. If X is a polyhedron, only its finitely many extreme points need to be included; in other cases X itself is finite, as in most combinatorial optimization problems. We will henceforth assume that the sums in (3) are in fact restricted to the proper finite set of relevant points in X, in order to restrict our treatment to finite LPs only.

After having reformulated (1) as (3), the idea of the DW algorithm is just that of *column generation (CG)*. A subset $\mathcal{B} \subseteq X$ is selected, and the corresponding *primal master problem*

$$\min_\theta \{ c\left(\textstyle\sum_{x \in \mathcal{B}} x\theta_x\right) \ : \ A\left(\textstyle\sum_{x \in \mathcal{B}} x\theta_x\right) = b \ , \ \theta \in \Theta \} \tag{4}$$

is solved; here, each (relevant) point $x \in X$ becomes a *column* in the LP (4), corresponding to the variable θ_x. Note that the "explicit form" (4) is equivalent to the "implicit form"

$$\min_x \{ cx : Ax = b \ , \ x \in X_{\mathcal{B}} = conv(\mathcal{B}) \} \ . \tag{5}$$

The primal master problem is therefore the *restriction* of (3) to the subset of columns $\mathcal{B}$ or, equivalently, an *inner linearizaton* of (1) where $conv(X)$ is substituted with the "simpler" set $conv(\mathcal{B})$. An optimal solution θ^* yelds a feasible solution $\tilde{x} = \sum_{x \in \mathcal{B}} x\theta^*_x$ for (1), and is typically accompanied by a dual optimal solution $(\tilde{v}, \tilde{\pi})$ for the *dual master problem*

$$\max_{v,\pi} \{ v \ : \ v \leq cx + \pi(b - Ax) \ , \ x \in \mathcal{B} \} \ . \tag{6}$$

It is now immediate to verify whether or not $\tilde{x}$ is optimal for (1) by checking if $(\tilde{v}, \tilde{\pi})$ is actually feasible for the dual of (3) (that is, (6) with $\mathcal{B} = X$). This can be done by solving the *pricing problem* $(P_{\tilde{\pi}})$, which determines the point (column) $\bar{x} \in X$ of least *reduced cost* $(c - \tilde{\pi}A)\bar{x}$. The complete Dantzig-Wolfe decomposition algorithm is summarized in the following pseudo-code.

```
⟨ initialize B ⟩;
do
   ⟨ solve (4) and (6) for x̃ and (ṽ, π̃), respectively ⟩;
   x̄ ∈ argmin_x { (c − π̃A)x : x ∈ X };          /* (P_π̃) */
   B = B ∪ {x̄};
while( ṽ > cx̄ + π̃(b − Ax̄) );
```

Figure 1: The Dantzig-Wolfe decomposition algorithm

We remark that, in order for the algorithm to be well-defined, it is necessary that the initial set $\mathcal{B}$ is "sufficiently large" to ensure that (6) has a finite optimal solution: however, we postpone the discussion to this point until Section 3.1. The DW algorithm has a "dual" interpretation in terms of Kelley's cutting-plane algorithm [24]: at each step, the dual master problem maximizes the *cutting-plane model*

$$f_{\mathcal{B}}(\pi) = \min_x \{ cx + \pi(b - Ax) : x \in \mathcal{B} \} \tag{7}$$

which is an *outer approximation* of the Lagrangian function $f(\pi)$ (i.e., $f_{\mathcal{B}} \geq f$). The set $\mathcal{B}$ can then be interpreted in dual terms as the *bundle* of available information that describes the Lagrangian function f (function value and first-order behavior) at some points in the space. Solving the pricing problem simply means evaluating f in $\tilde{\pi}$, and adding $\bar{x}$ to $\mathcal{B}$ means refining $f_{\mathcal{B}}$ until it is "exact" in some optimal point of (2). This dual interpretation allows to modify the standard DW approach in a number of ways aimed at improving its performances in practice, as discussed in Section 3.

2.1.1 The Noncompact Case

The DW algorithm can be easily extended to the case where X is noncompact, provided that the algorithm that solves the pricing problem is capable of providing an *unbounded descent direction* ν for $conv(X)$ whenever $f(\pi) = -\infty$. Since (P_π) is unbounded for any π such that $(c - \pi A)\nu < 0$, each such ν is associated with a linear constraint $(c - \pi A)\nu \geq 0$ that is valid for the *effective domain* π of $f(\pi)$ (the set of all points where it is finite-valued); in other words, the extreme rays of the *recession cone* $\mathcal{C}$ of $conv(X)$ characterize the effective domain of f. Then, one can replace $\mathcal{B}$ by $\mathcal{B}^0 \cup \mathcal{B}^1$, where $\mathcal{B}^0 \subseteq \mathcal{C}$ and $\mathcal{B}^1 \subseteq X$. The primal and dual master problem become, respectively

$$\min_\theta \begin{cases} c\left(\sum_{x\in\mathcal{B}^1} x\theta_x + \sum_{\nu\in\mathcal{B}^0} \nu\theta_\nu\right) \\ A\left(\sum_{x\in\mathcal{B}^1} x\theta_x + \sum_{\nu\in\mathcal{B}^0} \nu\theta_\nu\right) = b \\ \sum_{x\in\mathcal{B}^1} \theta_x = 1\ ,\ \theta \geq 0 \end{cases} \qquad \max_{v,\pi} \begin{cases} \pi b + v \\ v \leq (c - \pi A)x \quad x \in \mathcal{B}^1 \\ 0 \leq (c - \pi A)\nu \quad \nu \in \mathcal{B}^0 \end{cases} \tag{8}$$

or, equivalently,

$$\min_x \{\, cx : Ax = b\ ,\ x \in conv(\mathcal{B}^1) + cone(\mathcal{B}^0) \,\} \qquad \max_\pi \{\, f_{\mathcal{B}^1}(\pi) \ :\ \pi \in \pi_{\mathcal{B}} \,\}\ , \tag{9}$$

where $\pi_{\mathcal{B}} = \{\, \pi \ :\ (c - \pi A)\nu \geq 0\ ,\ \nu \in \mathcal{B}^0 \,\}$ is an outer approximation of the effective domain of f. At each iteration of the algorithm, the pricing problem computes either an optimal solution $\bar{x} \in X$, that is added to $\mathcal{B}^1$, or a feasible unbounded direction $\bar{\nu} \in \mathcal{C}$, that is added to $\mathcal{B}^0$. We do not go into further details of the convergence proofs of the DW algorithm, since they are subsumed by the theory of the SDW decomposition method that is developed next.

2.1.2 The Decomposable Case

The DW algorithm can also be specialized to the *decomposable* case, where $X = \mathsf{X}_{k\in K}\ X^k$ is the Cartesian product of $|K|$ sets (which we can assume to be compact, least applying the same approach as in Section 2.1.1 to each X^k). This means that the Lagrangian relaxation decomposes into k independent problems, i.e., any optimal solution $\bar{x}$ has the form $[\bar{x}^k]_{k\in K}$, where $\bar{x}^k$ is an optimal solution of the k-th subproblem. In other words, the Lagrangian function

$$f(\pi) = \pi b + \textstyle\sum_{k\in K} f^k(\pi)$$

is the sum of its k component functions

$$f^k(\pi) = \min_{x^k} \{\, (c^k - yA^k)x^k \ :\ x^k \in X^k \,\}\ .$$

In this case, one possibility is to solve at each iteration the *disaggregated primal master problem*

$$\min_\theta \begin{cases} \sum_{k\in K} c^k \sum_{x\in\mathcal{B}^k} x\theta_x \\ \sum_{k\in K} A^k \sum_{x\in\mathcal{B}^k} x\theta_x = b \\ \sum_{x\in\mathcal{B}^k} \theta_x = 1 \quad k \in K\ ,\ \theta \geq 0 \end{cases} \tag{10}$$

instead of (4). Each component x^k of each of the solutions generated so far in the process is kept in the set $\mathcal{B}^k$ and has an associated convex multiplier θ_{x^k}, independent from the multiplier associated to other components belonging to the same solution. The corresponding *disaggregated dual master problem* is

$$\max_{v,\pi} \left\{\, \pi b + \textstyle\sum_{h\in K} v^k \ :\ v^k \leq (c^k - \pi A^k)x^k \quad x^k \in \mathcal{B}^k \quad k \in K \,\right\} \ =\ \max_\pi \left\{\, \textstyle\sum_{k\in K} f^k_{\mathcal{B}}(\pi) \,\right\}\ ,$$

where $f_{\mathcal{B}}^k = f_{\mathcal{B}^k}$ is the cutting-plane model of the k-th component f^k of f. Analogously, (10) can be rewritten as

$$\min_x \left\{ \sum_{k\in K} c^k x^k \ : \ \sum_{k\in K} A^k x^k = b \ , \ x^k \in X_{\mathcal{B}}^k = conv(\mathcal{B}^k) \quad k \in K \right\}$$

where $X_{\mathcal{B}}^k$ is an inner approximation of $conv(X^k)$. It is easy to see that, for the same set of Lagrangian solutions $\mathcal{B} \subseteq X$, the feasible set of (10) strictly contains that of (5); in fact, (4) is the restriction of (10) where all the components $\bar{x}^k$ corresponding to the same solution $\bar{x}$ must have the same convex multiplier. In other words, $\mathsf{X}_{k\in K}\ conv(\mathcal{B}^k)$ is a better approximation of $conv(X)$ than $conv(\mathcal{B})$; equivalently, the sum of the separate cutting-plane models $f_{\mathcal{B}}^k$ is a better approximation of f than the "aggregated" cutting-plane model $f_{\mathcal{B}}$. The tradeoff is that the disaggregated master problems are roughly $|K|$ times larger than the corresponding aggregated ones, and therefore they are more costly to solve; however, they use the available information more efficiently, which often results in a much faster convergence, and ultimately in better overall performances [23].

2.2 Structured Dantzig-Wolfe Decomposition

The SDW method can be applied to solving problem (1) provided that the same assumptions than in the DW approach hold, and some extra knowledge about the structure of $conv(X)$ is available.

Assumption 1 *For a finite vector of variables θ and matrices C, Γ and γ of appropriate dimension,*

$$conv(X) = \{ \ x = C\theta \ : \ \Gamma\theta \leq \gamma \ \} \ .$$

In other words, one needs a proper *reformulation* of $conv(X)$ in a different space of variables θ. Of course, we assume θ to be "large" but amenable to solution through variable restriction. That is, let $\mathcal{B}$ be any subset of the index set of the variables θ; we denote by $\theta_{\mathcal{B}}$ the corresponding subvector of variables, and with $\Gamma_{\mathcal{B}}$, $\gamma_{\mathcal{B}}$, respectively, the coefficient matrix and the right-hand side of all (and only) the constraints concerning variables with indices in $\mathcal{B}$. Analogously, we denote by $C_{\mathcal{B}}$ the restriction of the matrix (linear mapping) C to the indices in $\mathcal{B}$. With this notation, we introduce a further assumption.

Assumption 2

$$\Gamma_{\mathcal{B}}\bar{\theta}_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \ \Rightarrow \Gamma \begin{bmatrix} \bar{\theta}_{\mathcal{B}} \\ 0 \end{bmatrix} \leq \gamma \ .$$

This assumption simply means that we can always "pad" with zeroes a partial solution without fear of losing feasibility, even if we do not know "many" constraints of the reformulation; this is clearly necessary in order to apply a variable/constraint generation procedure. Hence, by denoting

$$X_{\mathcal{B}} = \{ \ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ : \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \ \}$$

we always have that $X_{\mathcal{B}} \subseteq conv(X)$. In other words, the function

$$f_{\mathcal{B}}(\pi) = \min_x \{ \ cx + \pi(b - Ax) \ : \ x \in X_{\mathcal{B}} \ \} \tag{11}$$

is still an "outer approximation" of the Lagrangian function f, in the sense that $f_{\mathcal{B}} \geq f$. We will therefore keep the name "bundle" for $\mathcal{B}$, since it still can be interpreted as a subset of the whole information required to describe the behavior of f.

Finally, we need to assume that, given a solution that *cannot* be expressed by means of a given index set $\mathcal{B}$, it must be easy to update the index set and the partial description of the constraints in order to incorporate at least some of the information provided by that solution.

Assumption 3 *Let $\mathcal{B}$ be an index set of θ and let $\bar{x}$ be a point such that $\bar{x} \in conv(X) \setminus X_{\mathcal{B}}$; then, it must be easy to update $\mathcal{B}$ and the associated partial description of the constraints $\Gamma_{\mathcal{B}}$, $\gamma_{\mathcal{B}}$ to a set $\mathcal{B}' \supset \mathcal{B}$ such that there exists $\mathcal{B}'' \supseteq \mathcal{B}'$ with $\bar{x} \in X_{\mathcal{B}''}$.*

This assumption is purposely stated in a rather abstract form. What one expects is that given a solution $\bar{x} \notin X_{\mathcal{B}}$, it should be relatively easy to find out "why" it is not feasible; that is, to find at least one variable that is not present in the description of $X_{\mathcal{B}}$, and that is necessary to represent the given solution $\bar{x}$. A nontrivial example of such a procedure is described in Section 4, and other cases can be easily devised. For instance, the well-known Gilmore-Gomory formulation of the cutting stock problem [6] is usually solved by DW/CG approaches; there, X is the set of all valid *cutting patterns*, that is, all feasible solutions to an integer knapsack problem. Owing to the well-known reformulation of integer knapsack problems in terms of longest path problems on a directed acyclic network, one can devise the *arc-flow model* of the cutting stock problem [4], that provides the same lower bound at the cost of a pseudo-polynomial number of variables (one for each arc in the graph representing the knapsack) and constraints (one for each node in the same graph). Thus, the arc-flow model provides the alternative reformulation of Assumption 1, whose pricing problem is an integer knapsack as in the standard DW decomposition applied to the Gilmore and Gomory model. Each feasible cutting pattern produced by the pricing problem is a path in the directed graph underlying the arc-flow model. Thus, one can easily devise a restricted formulation $X_{\mathcal{B}}$ corresponding to a (small) sub-graph of the (large) full directed graph; each time a new cutting pattern $\bar{x}$ is generated, it is easy to check whether or not it corresponds to a path in the current subgraph. If not, at least one new arc (and, possibly, node) can be identified to produce $\mathcal{B}'$.

It is now immediate to extend the DW approach. The primal master problem becomes, in "explicit" form

$$\min_{x,\theta} \{ cx : Ax = b , x = C_{\mathcal{B}}\theta_{\mathcal{B}} , \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \} . \tag{12}$$

This has the same "implicit form" (5) as the standard DW, except that $\mathcal{B}$ does not (necessarily) denote a set of extreme points of X, and $X_{\mathcal{B}}$ is therefore not (necessarily) their convex hull. The *Structured Dantzig-Wolfe* decomposition method is then described in the following pseudo-code.

```
⟨ initialize B ⟩;
repeat
   ⟨ solve (12) for x̃; let ṽ = cx̃ and π̃ be optimal dual multipliers of Ax = b ⟩;
   x̄ ∈ argmin_x { (c − π̃A)x : x ∈ X };                    /* (P_π̃) */
   if( ṽ = cx̄ + π̃(b − Ax̄) )
      then STOP;                                          /* x̃ optimal */
      else ⟨ update B as in Assumption 3 ⟩;
until ~ STOP
```

Figure 2: The Structured Dantzig-Wolfe decomposition algorithm

We now turn to proving finiteness and correctness of the proposed algorithm. While this may be deemed not strictly necessary, in view of the convergence theory available for the more sophisticated *Stabilized* versions (cf. §3), it is useful to emphasize the role of Assumptions 1, 2 and 3. We start assuming that

- X is compact;
- at the first iteration (and, therefore, at all the following ones) (12) has a feasible solution;

but we later show how to remove these assumptions.

Lemma 4 *At all iterations of the SDW algorithm we have $\tilde{v} \geq c\bar{x} + \tilde{\pi}(b - A\bar{x})$; if equality holds then $\tilde{x}$ is optimal for (1).*

Proof. Due to Assumption 2, (12) is a restriction of (1) while $(P_{\tilde{\pi}})$ is a relaxation of (1). Therefore $c\bar{x} + \tilde{\pi}(b - A\bar{x}) = v(P_{\tilde{\pi}}) \leq v(1) \leq v(12) = \tilde{v}$. ■

Lemma 5 *At all iterations where the SDW algorithm does not stop, for any optimal $\bar{x}$ for $(P_{\tilde{\pi}})$ and any $\bar{\theta}$ such that $\bar{x} = C\bar{\theta}$ there must be at least one nonzero variable in $\bar{\theta}$ whose index does not belong to $\mathcal{B}$.*

Proof. It is well-known that any optimal solution $\tilde{x}$ of (12) is also an optimal solution of

$$\min_{x,\theta} \{ cx + \tilde{\pi}(b - Ax) \ : \ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ , \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \} \tag{13}$$

since $\tilde{\pi}$ are optimal dual multipliers for the constraints $Ax = b$ in (12). It is also clear that $\tilde{v} = c\tilde{x} = v(13)$. Now, assume by contradiction that there is an optimal solution $\bar{x}$ for $(P_{\tilde{\pi}})$ and a $\bar{\theta}$ such that $\bar{x} = C\bar{\theta}$, and all the nonzeroes in $\bar{\theta}$ belong to $\mathcal{B}$. Then, $[\bar{x}, \bar{\theta}]$ is a feasible solution of (13) with cost $c\bar{x} + \tilde{\pi}(b - A\bar{x}) < \tilde{v} = c\tilde{x}$, contradicting optimality of $\tilde{x}$. ■

Theorem 6 *The SDW algorithm finitely terminates with an optimal solution of (1).*

Proof. By Lemma 4, if the algorithm terminates then it provides an optimal solution of (1). By Lemma 5, at each iteration where the algorithm does not terminate the solution $\bar{x}$ of the pricing problem "points out" at least one variable whose index does not currently belong to $\mathcal{B}$. Thus, by Assumption 3, the size of $\mathcal{B}$ increases by at least one. Hence, the algorithm terminates in a finite number of iterations. ■

The convergence of the algorithm can be ensured even if (proper) removal of indices from $\mathcal{B}$ is allowed.

Theorem 7 *Let $\tilde{v}_k$ be the optimal objective function value of (12) at iteration k; the SDW algorithm finitely terminates with an optimal solution of (1) even if, at each iteration k such that $\tilde{v}_k < \tilde{v}_{k-1}$, all the indices of variables in $\mathcal{B}$ that have zero value in the current optimal solution of (12) are removed from $\mathcal{B}$.*

Proof. By the above analysis, it is clear that $\{\tilde{v}_k\}$ is a nonincreasing sequence. Hence, no two iterations $h > k$ can have the same $\mathcal{B}$: either $\tilde{v}_h < \tilde{v}_k$, and therefore $\mathcal{B}$ must be different, or $\tilde{v}_h = \tilde{v}_k$, and therefore the size of $\mathcal{B}$ must have strictly increased. Therefore, only a finite number of iterations can be performed. ■

Clearly, the SDW algorithm generalizes the original DW approach. In fact, the reformulation used in the DW approach is

$$conv(X) = \{ \textstyle\sum_{x \in X} x\theta_x \ : \ \theta \in \Theta \} \ .$$

In other words, C is the matrix having as columns the (extreme) points of X, and Γ, γ just describe the unitary simplex Θ. Clearly, Assumptions 2 and 3 are satisfied; for the latter, in particular, the optimal solution $\bar{x}$ of the pricing problem uniquely identifies the variable $\theta_{\bar{x}}$ that has to be nonzero in order to be able to comprise $\bar{x}$ into $X_{\mathcal{B}}$ (the variables are indexed by the points in X). The

generality of the DW approach stems from the fact that is it always possible to express $conv(X)$ by means of the (extreme) points of X. However, the SDW approach allows to exploit any other known reformulation of $conv(X)$. As shown in Section 4.5, this may improve the performances of the approach with respect to the standard reformulation used in the DW method.

As for the original DW approach (cf. §2.1.1), the SDW algorithm can be easily extended to the case where X is not compact, provided that, as usual, the algorithm that solves the pricing problem is capable of providing an unbounded ascent direction ν for $conv(X)$ whenever $f(\pi) = -\infty$, and that the following strengthened form of Assumption 3 holds.

Assumption 8 *Let Assumption 3 hold. Furthermore, let $\mathcal{B}$ be an index set of θ and let ν be an unbounded ascent direction for $conv(X)$ which is* not *an unbounded ascent direction for $X_{\mathcal{B}}$. Then, it must be easy to update $\mathcal{B}$ and the associated partial description of the constraints $\Gamma_{\mathcal{B}}$, $\gamma_{\mathcal{B}}$ to a $\mathcal{B}' \supset \mathcal{B}$ such that there exists a $\mathcal{B}'' \supseteq \mathcal{B}'$ such that ν is an unbounded ascent direction for $X_{\mathcal{B}''}$.*

Under these assumptions, the SDW method extends directly to the non-compact case. Indeed, the form of the primal master problem does not even change: if an unbounded ascent direction ν is produced by the pricing problem, then it is used to update $\mathcal{B}$ exactly as any point $\bar{x}$. Note that, in order for (12) to be able to represent a noncompact set, some of the θ variables must not be bounded to lie in a compact set, see e.g. (8).

The SDW approach that we have presented and analyzed generalizes the DW decomposition method by allowing to exploit available information about the structure of the "easy" set X which would otherwise go unnoticed. The basic ingredients of the approach are the same as in the DW approach: *reformulation* and *variables generation*. For DW, however, the reformulation part is standard, and therefore tends to be overlooked; this is not the case for the SDW approach.

3 Stabilizing Structured Dantzig-Wolfe Decomposition

3.1 Issues in the SDW Approach

The SDW approach in the above form is simple to describe and, given the availability of efficient and well-engineered linear optimization software, straightforward to implement. However, several nontrivial issues have to be addressed.

Empty master problem. In order to be well-defined, the DW method needs a starting $\mathcal{B}$ such that (4) has a finite optimal solution. There are two basic strategies for ensuring this. One is having $\hat{x} \in \mathcal{B}$ where $\hat{x}$ is a feasible solution for (1) (e.g., a feasible solution to (P)), which corresponds to inserting the constraint $v \leq c\hat{x}$ into (6). Alternatively, a standard "Phase 0" approach can be implemented where

$$\min_{x,\theta} \left\{ \left|\left| A\left(\sum_{x \in \mathcal{B}} x\theta_x\right) - b \right|\right|_\infty \; : \; \theta \in \Theta \right\}$$

(and the corresponding dual) is solved at each iteration instead of (3) until a $\mathcal{B}$ which provides a feasible solution is found, or (1) is proved to be unfeasible. Similar ideas work for the SDW method. If some $\hat{x}$ is available, a $\mathcal{B}$ can be (iteratively applying Assumption 3 if necessary) determined which "entirely covers" it, i.e., allow to have it expressed as a feasible solution to (12). If that is not possible (e.g., (1) is not known a priori to have a feasible solution), a "Phase 0" approach can be used where we initially consider

$$\min_x \left\{ ||Ax - b||_\infty \; : \; x \in conv(X) \right\} \tag{14}$$

instead of (1). Since $v(14) = 0$ if and only if (1) has a feasible solution, we apply the SDW approach to (14) until either a feasible solution of (1)—more to the point, a $\mathcal{B}$ such that (12) has a feasible solution—is found, or it is proven that $v(14) > 0$. In order to apply the SDW approach to the solution of (14), the primal master problem becomes

$$\min_{x,\theta} \{ \|Ax - b\|_\infty : x = C_{\mathcal{B}}\theta_{\mathcal{B}} , \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \} \tag{15}$$

or, more explicitly,

$$\min_{w,x,\theta} \{ w : -we \leq Ax - b \leq we , x = C_{\mathcal{B}}\theta_{\mathcal{B}} , \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \} \tag{16}$$

where e is the vector of all ones of proper dimension. The pricing problem (the Lagrangian relaxation with respect to the constraints $-we \leq Ax - b \leq we$) is

$$\min_x \{ (\pi^+ - \pi^-)(b - Ax) : x \in X \} \tag{17}$$

where π^+ and π^+ are nonnegative Lagrangian multipliers such that $\pi^+e + \pi^-e = 1$, which is a constraint of the dual of (16), and therefore is automatically satisfied by the dual solutions used in the SDW algorithm (without this constraint, the "true" pricing problem would be unbounded). It is now immediate to mirror the proofs of Lemmas 4 and 5 to obtain once again Theorems 6 and 7. In fact, once again (16) is a restriction of (14) while (17) is a relaxation of (14). Thus, we can easily implement a "Phase 0" of the SDW algorithm which finds a proper $\mathcal{B}$ set for starting the standard SDW approach. The only standing assumption is that we must be able to easily find a set $\mathcal{B}$ such that (16) has a feasible solution, that is, the set $\Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}$ is nonempty; due to Assumption 2, this is most likely to be immediate (it is in the original DW approach). As we shall see soon, *stabilized* versions of both the DW and the SDW algorithms are build upon this idea, hence they do not need a Phase 0.

Instability. The above discussion may have mislead the reader in believing that generating a good approximation of the dual optimal solution π^* is enough to solve the problem; unfortunately, this is far from being true. Quite surprisingly, even generating the initial $\mathcal{B}$ by solving a subproblem at a very good approximation of π^* may have little effect on the convergence of the DW algorithm. The issue here (see for instance [5, 17]) is that there is no control over the "oracle" which computes $f(\pi)$: even if it is called at the very optimal point π^*, it would not in general report the *whole* set $\mathcal{B}^*$ of points that are necessary to *prove* its optimality. Thus, one would expect a "smart" algorithm, provided with knowledge about π^*, to sample the dual space "near" the dual optimal solution in order to force the subproblem to generate other points that are also optimal for π^*. However, there is nothing in the DW algorithm that may generate this behavior: the next $\tilde{\pi}$ is simply chosen as the maximizer of $f_{\mathcal{B}}$. The sequence of dual solutions $\{\tilde{\pi}\}$ has therefore no "locality" properties: even if a good approximation of π^* is obtained at some iteration, the dual solution at the subsequent iteration may be arbitrarily far from optimal. In other words, the DW approach is almost completely incapable of exploiting the fact that it has already reached a good dual solution in order to speed up the subsequent computations. This is known as the *instability* of the approach, which is the main cause of its slow convergence rate on many practical problems, and clearly applies to the SDW approach as well.

All this suggests to introduce some mean to "stabilize" the sequence of dual iterations. If π^* were actually known, one may simply restrict the dual iterates in a small region surrounding it, forcing the subproblem to generate points that are "almost optimal" in π^* and, consequently, efficiently accumulate $\mathcal{B}^*$. This would actually be very efficient [5], but in general π^* is *not* known. A possible solution is to use an estimate instead, taking into account the case where the estimate of the dual optimal solution is not exact.

3.2 A Stabilized SDW Approach

To stabilize the SDW approach, we exploit some ideas originally developed in the field of nondifferentiable optimization. In order to avoid large fluctuations of the dual multipliers, a "stabilization device" is introduced in the dual problem; this is done by choosing a *current center* $\bar{\pi}$, a family of proper convex *stabilizing functions* $\mathcal{D}_t : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ dependent on a real parameter $t > 0$, and by solving the *stabilized dual master problem*

$$\max_\pi \{ f_{\mathcal{B}}(\pi) - \mathcal{D}_t(\pi - \bar{\pi}) \} \tag{18}$$

instead of (6) at each iteration. Constraints $\pi \in \pi_{\mathcal{B}}$ in (18) (see (9)) can be taken into account at the cost of some notational complication, which is avoided here in order to keep the presentation simpler. The optimal solution $\tilde{\pi}$ of (18) is then used to compute $f(\tilde{\pi})$ as in the standard scheme. The stabilizing function $\mathcal{D}_t$ is meant to penalize points "too far" from $\bar{\pi}$; at a first reading a norm-like function can be imagined there, with more details due soon. It should be noted that other, more or less closely related, ways for stabilizing DW/CG algorithms have been proposed; a thorough discussion of the relationships among them can be found in [22, 25].

Solving (18) is equivalent to solving a *generalized augmented Lagrangian* of $(P_{\mathcal{B}})$, using as augmenting function the *Fenchel conjugate* of $\mathcal{D}_t$. For any convex function $f(\pi)$, its Fenchel conjugate

$$f^*(z) = \sup_\pi \{ z\pi - f(\pi) \}$$

characterizes the set of all vectors z that are support hyperplanes to the epigraph of f at some point. f^* is a closed convex function and features several properties [16, 21]; here we just remind that from the definition $f^*(0) = -\inf_\pi \{ f(\pi) \}$. In our case, we can reproduce the well-known result that *the Fenchel conjugate of a Lagrangian function is the value function of the original problem*. In fact,

$$(-f_{\mathcal{B}})^*(z) = \sup_\pi \{ z\pi + f_{\mathcal{B}}(\pi) \} = \sup_\pi \{ \min_x \{ cx + \pi(z + b - Ax) : x \in X_{\mathcal{B}} \} \}$$

which can be described by saying that $(-f_{\mathcal{B}})^*(z)$ is the optimal value of the Lagrangian Dual of

$$\min_x \{ cx : z = Ax - b , x \in X_{\mathcal{B}} \}$$

with respect to the "perturbed" $z = Ax - b$ constraints. The optimal value of the above problem, as a function of z, is known as the *value function* of the master problem (5); standard results in Lagrangian Duality [21] guarantee that the above problem is equivalent to its Lagrangian Dual, i.e., that

$$(-f_{\mathcal{B}})^*(z) = \min_x \{ cx : z = Ax - b , x \in X_{\mathcal{B}} \} . \tag{19}$$

We can then compute the *Fenchel dual* [21] of (18), which is obtained by simply evaluating the Fenchel conjugate of the objective function at zero; after some algebra, this can be shown to result in

$$\min_z \{ (-f_{\mathcal{B}})^*(z) - \bar{\pi}z + \mathcal{D}_t^*(-z) \}$$

which plugging in the definition of $f_{\mathcal{B}}^*$ gives

$$\min_{z,x} \{ cx - \bar{\pi}z + \mathcal{D}_t^*(-z) : z = Ax - b , x \in X_{\mathcal{B}} \} . \tag{20}$$

The *stabilized primal master problem* (20) is equivalent to (18); indeed, we can assume that whichever of the two is solved, both an optimal primal solution $(\tilde{x}, \tilde{z})$ and an optimal dual solution $\tilde{\pi}$ are simultaneously computed. For practical purposes, one would more likely implement (20) rather than (18); however, the following proposition shows how to recover all the necessary dual information once it is solved.

Theorem 9 *Let $(\tilde{x}, \tilde{z})$ be an optimal primal solution to (20) and $\tilde{\pi}$ be optimal Lagrangian (dual) multipliers associated to constraints $z = Ax - b$; then, $\tilde{\pi}$ is an optimal solution to (18), and $f_{\mathcal{B}}(\tilde{\pi}) = c\tilde{x} + \tilde{\pi}(b - A\tilde{x})$.*

Proof. Because $\tilde{\pi}$ are optimal dual multipliers, $(\tilde{x}, \tilde{z})$ is an optimal solution to

$$\min_{z,x} \{ cx - \bar{\pi}z + \mathcal{D}_t^*(-z) + \tilde{\pi}(z - Ax + b) \ : \ x \in X_{\mathcal{B}} \} \ .$$

That is,

$$\tilde{z} \in \operatorname{argmin}_z \{ (\tilde{\pi} - \bar{\pi})z + \mathcal{D}_t^*(-z) \}$$

which is equivalent to $0 \in \partial[(\tilde{\pi} - \bar{\pi}) \cdot + \mathcal{D}_t^*(-\cdot)](\tilde{z})$, which translates to $0 \in \{\tilde{y} - \bar{y}\} - \partial\mathcal{D}_t^*(-\tilde{z})$, which finally yields $\tilde{\pi} - \bar{\pi} \in \partial\mathcal{D}_t^*(-\tilde{z})$. Furthermore, by

$$\tilde{x} \in \operatorname{argmin}_x \{ cx + \tilde{\pi}(b - Ax) \ : \ x \in X_{\mathcal{B}} \}$$

one first has $f_{\mathcal{B}}(\tilde{\pi}) = c\tilde{x} + \tilde{\pi}(b - A\tilde{x})$ as desired. Then, because $\tilde{x}$ is a minimizer, $b - A\tilde{x}$ is a *supergradient* of the *concave* function $f_{\mathcal{B}}$; changing sign, $-(b - A\tilde{x}) = \tilde{z} \in \partial(-f_{\mathcal{B}})(\tilde{\pi})$. The desired results now follow from [16, Lemma 2.2, conditions (2.2) and (2.3)] after minor notational adjustments ($-f_{\mathcal{B}}$ in place of $f_{\mathcal{B}}$, $\tilde{\pi} - \bar{\pi}$ in place of d^*). ■

Using $\mathcal{D}_t = \frac{1}{2t}\|\cdot\|_2^2$, which gives $\mathcal{D}_t^* = \frac{1}{2}t\|\cdot\|_2^2$, one immediately recognizes in (20) the *augmented Lagrangian* of (4), with a "first-order" Lagrangian term corresponding to the current point $\bar{\pi}$ and a "second-order" term corresponding to the stabilizing function $\mathcal{D}_t$. Using a different stabilizing term $\mathcal{D}_t$ in the dual corresponds to a nonquadratic augmented Lagrangian. Note that the "null" stabilizing term $\mathcal{D}_t = 0$ corresponds to $\mathcal{D}_t^* = I_{\{0\}}$; that is, with no stabilization at all, (20) collapses back to (5)/(12). This is the extreme case of a general property, that can be easily checked for varying t in the quadratic case; as $f \leq g \Rightarrow f^* \geq g^*$, a "flatter" $\mathcal{D}_t$ in the dual corresponds to a "steeper" $\mathcal{D}_t^*$ in the primal, and vice-versa. Also, note that the above formulae work for $X_{\mathcal{B}} = X$ as well, i.e., for the original problems (1)/(2) rather than for their approximations.

The stabilized master problems provide means for defining a general Stabilized Structured Dantzig-Wolfe algorithm (S^2DW), such as that of Figure 3.

```
⟨ Initialize π̄ and t ⟩
⟨ solve P_π̄, initialize B with the resulting x̄ ⟩
repeat
   ⟨ solve (20) for x̃; let π̃ be optimal dual multipliers of z = Ax − b ⟩;
   if( cx̃ = f(π̄) && Ax̃ = b )
      then STOP;                                          /* x̄ optimal */
      else
         x̄ ∈ argmin_x { (c − π̃A)x : x ∈ X }; f(π̃) = cx̄ + ỹ(b − Ax̄);    /* (P_π̃) */
         ⟨ update B as in Assumption 3 ⟩;
         if( f(π̃) is "substantially better" than f(π̄) )
         then π̄ = π̃                                       /*Serious Step*/
         ⟨ possibly update t ⟩
until ~ STOP
```

Figure 3: The Stabilized Structured Dantzig-Wolfe algorithm

The algorithm generates at each iteration a *tentative point* $\tilde{\pi}$ for the dual and a (possibly unfeasible) primal solution $\tilde{x}$ by solving (20). If $\tilde{x}$ is feasible and has a cost equal to the lower bound $f(\bar{\pi})$,

then it is clearly an optimal solution for (1), and $\bar{\pi}$ is an optimal solution for (2). More in general, one can stop the algorithm when $c\tilde{x} + \bar{\pi}(b - A\tilde{x}) - f(\bar{\pi}) \geq 0$ and $||A\tilde{x} - b||$ (with any norm) are both "small" numbers. Otherwise, new elements of $\mathcal{B}$ are generated using the tentative point $\tilde{\pi}$.

If $f(\tilde{\pi})$ is "substantially better" than $f(\bar{\pi})$, then it is worth to update the current center: this is called a *Serious Step (SS)*. Otherwise, the current center is not changed, and we rely on the fact that $\mathcal{B}$ is improved for producing, at the next iteration, a better tentative point $\tilde{\pi}$: this is called a *Null Step (NS)*. In either case, the stabilizing term can be changed, usually in different ways according to the outcome of the iteration. If a *SS* is performed, i.e., the current approximation $f_{\mathcal{B}}$ of f was able to identify a point $\tilde{\pi}$ with better function value than $\bar{\pi}$, then it may be worth to "trust" the model more and lessen the penalty for moving far from $\bar{\pi}$; this corresponds to a "steeper" penalty term in the primal. Conversely, a "bad" *NS* might be due to an excessive trust in the model, i.e., an insufficient stabilization, thereby suggesting to "steepen" $\mathcal{D}_t$ ($\Rightarrow$ "flatten" $\mathcal{D}_t^*$).

When $f_{\mathcal{B}}$ is the standard cutting-plane model (7), the above approach is *exactly* what is usually referred to as a (generalized) bundle method [16]; thus, S^{2}DW is a bundle method where the model $f_{\mathcal{B}}$ is "nonstandard". It is worth remarking that the ordinary bundle method, just like the DW/CG approach of which it is a variant, exists in the disaggregated variant for the decomposable case $X = \bigtimes_{k \in K} X^k \Rightarrow f(\pi) = \pi b + \sum_{k \in K} f^k(\pi)$ (see Section 2.1.2); in that case, too, $f_{\mathcal{B}}$ is not (exactly) the "standard" cutting-plane model (but rather the sum of standard cutting-plane models). And in that case, too, the extra cost for the $|K|$ times larger master problems is often largely compensated by a much faster convergence which leads to better overall performances [2, 7]. Thus, S^{2}DW can be seen as only carrying this idea to its natural extension by using an "even more disaggregated" (specialized) model in place of the standard cutting-plane one.

3.3 Convergence Conditions

The algorithm in Figure 3 can be shown to finitely converge to a pair (π^*, x^*) of optimal solutions to (2) and (1), respectively, under a number of different hypotheses. Within our environment, the following conditions can be imposed:

i) $\mathcal{D}_t$ is a convex nonnegative function such that $\mathcal{D}_t(0) = 0$, its level sets $S_\delta(\mathcal{D}_t)$ are *compact and full-dimensional* for all $\delta > 0$; remarkably, these requirements are *symmetric* in the primal, i.e., they hold for $\mathcal{D}_t^*$ *if and only if* they hold for $\mathcal{D}_t$ [16].

ii) $\mathcal{D}_t$ is *differentiable* in 0 and *strongly coercive*, i.e., $\lim_{\|\pi\| \to \infty} D_t(\pi)/\|\pi\| = +\infty$; equivalently, $\mathcal{D}_t^*$ is *strictly convex* in 0 and *finite everywhere*.

iii) A *necessary* condition to declare $f(\tilde{\pi})$ "substantially higher" than $f(\bar{\pi})$ is

$$f(\tilde{\pi}) - f(\bar{\pi}) \geq m(f_{\mathcal{B}}(\tilde{\pi}) - f(\bar{\pi})) \tag{21}$$

for some fixed $m \in (0, 1]$. A SS may not be performed even if (21) holds, but this can happen only finitely many times: during a sequence of *consecutive NSs*, at length (21) is also sufficient.

iv) During a sequence of *consecutive NSs*, $f(\tilde{\pi})$ is computed and at least one of the corresponding items is inserted into $\mathcal{B}$ at all iterations *but* possibly finitely many ones.

v) During a sequence of *consecutive NSs*, t can change only *finitely many times*.

vi) D_t is nonincreasing as a function of t, and $\lim_{t\to\infty} D_t(\pi) = 0$ for all π, i.e., it converges pointwise to the constant zero function. Dually, D_t^* is nondecreasing as a function of t and converges pointwise to $I_{\{0\}}$.

Under the above assumptions, global convergence of S^{2}DW can be proven using the results of [16]. This is easier with the further assumption

vii) $\mathcal{D}_t$ is *strictly convex*; equivalently, $\mathcal{D}_t^*$ is *differentiable*,

that is satisfied e.g. by the classic $\mathcal{D}_t = \frac{1}{2t}\|\cdot\|_2^2$ (but not by other useful stabilizing terms, see Section 3.5). The only delicate point is the treatment of $\mathcal{B}$ along the iterations. In fact, the theory in [16] does not mandate any specific choice for the model, thereby allowing the use of (11), provided that:

- $f_{\mathcal{B}}$ is a model of f in the sense that $f_{\mathcal{B}} \geq f$, which is precisely what Assumption 2 dictates (the requirement is $f_{\mathcal{B}} \leq f$ in [16] since it is written for minimization of convex functions);
- The bundle $\mathcal{B}$ is dealt with with minimal care. The handling of $\mathcal{B}$ is termed "β-strategy" in [16], and the basic requirement is that it is *monotone* [16, Definition 4.6]: this means that *at length, during a sequence of consecutive NSs*, one must have

$$(-f_{\mathcal{B}_{i+1}})^*(\tilde{z}_i) \leq (-f_{\mathcal{B}_i})^*(\tilde{z}_i) \tag{22}$$

  where $\mathcal{B}_i$ is the bundle at the i-th iteration of the sequence, and $\tilde{z}_i$ is the optimal solution (in the space of "constraints violation") of the stabilized primal master problem (20).

From (19), it is clear that a monotone β-strategy can be obtained with some sort of monotonicity in $\mathcal{B}$; for instance, if nothing is ever removed from it, then monotonicity is ensured. One can do better, allowing removal of items from $\mathcal{B}$ as long as "the important ones" are left in.

Lemma 10 *Assume that for any sequence of NSs, at length the bundle $\mathcal{B}_i$ has the following property: the optimal solution $\tilde{x}_i$ at step i is still a feasible solution to (20) at step $i+1$, i.e., with bundle $\mathcal{B}_{i+1}$. Then, (22) holds.*

Proof. Obvious, since $\tilde{z}_i = b - A\tilde{x}_i$ is uniquely identified by $\tilde{x}_i$. ■

By the above Lemma, it is basically only necessary to look, at each iterations, to the optimal value $\theta_{\mathcal{B}}^*$ of the "auxiliary" variables in the current definition of $X_{\mathcal{B}}$ (this is clearly available after having solved (20)); all the variables with zero value can be discarded. As we will see, a monotone β-strategy suffices under assumption vii); however, we may want to do without. This requires a slightly "stronger grip" on $\mathcal{B}$; one possibility is an *asymptotically blocked* β-strategy such that, for any sequence of NSs, at length removals from the bundle $\mathcal{B}$ are inhibited. Clearly, an asymptotically blocked β-strategy is monotone, and the notion is weaker than that of *strictly monotone* strategy [16, Definition 4.8].

An interesting point to discuss is that of *aggregation*. In general, monotonicity only requires that the optimal solution to the stabilized dual master problem $\tilde{z}$ remains feasible at the subsequent iteration. When using the standard cutting-plane model (7), there is a particularly simple way of attaining it: it suffices to *add $\tilde{x}$ to $\mathcal{B}$*, possibly *removing every other point*. Indeed, the linear function

$$f_{\tilde{x}}(\pi) = c\tilde{x} + \pi(b - A\tilde{x})$$

is a model of f, since $\tilde{x} \in X_{\mathcal{B}} \subseteq X$, and therefore $\tilde{z} = b - A\tilde{x}$ is feasible to (20). Thus, even resorting to the "poorman cutting-plane model" $f_{\tilde{x}}$ is sufficient to attain a convergent approach under some

conditions (cf. Lemma 11). This basically makes the algorithm a subgradient-type approach with deflection [3, 11], and may result in much slower convergence [8, 20].

While such a harsh approximation of the objective function is contrary in spirit to the S^{2}DW approach, where one precisely wants to obtain a more accurate model, it may be interesting to note that, under mild conditions, performing aggregation is possible even with a different model than the cutting-plane one. The idea is to take the alternative model (11) and consider

$$\bar{f}_{\mathcal{B}} \;=\; \min\{\, f_{\mathcal{B}} \,,\; f_{\tilde{x}} \,\}.$$

This clearly is again a model: $f \leq f_{\tilde{x}}$ and $f \leq f_{\mathcal{B}}$ imply $f \leq \bar{f}_{\mathcal{B}}$, and the model is "at least as accurate" as $f_{\mathcal{B}}$. One then wants to compute $(-\bar{f}_{\mathcal{B}})^*$, i.e., the conjugate of $\max\{\, -f_{\mathcal{B}} \,,\; -f_{\tilde{x}} \,\}$; a little conjugacy calculus (the results of [21] being more than sufficient) gives

$$epi\,(-\bar{f}_{\mathcal{B}})^* \;=\; cl\; conv(\; epi\,(-f_{\mathcal{B}})^* \,,\; epi\,(-f_{\tilde{x}})^* \;),$$

where it is easy to verify that

$$(-f_{\tilde{x}})^*(z) = \left\{ \begin{array}{ll} c\tilde{x} & \text{if } z = A\tilde{x} - b \\ +\infty & \text{otherwise.} \end{array} \right. .$$

Thus, the epigraph of $(-\bar{f}_{\mathcal{B}})^*$ can be constructed by taking all points (in the epigraphical space) $(z', (-f_{\mathcal{B}})^*(z'))$ and computing their convex hull with the single point $(A\tilde{x} - b, c\tilde{x})$, as $z'' = A\tilde{x} - b$ is the only point at which $(-f_{\tilde{x}})^*$ is not $+\infty$; the function value is then the inf over all these possibilities for a fixed z (the closure operation is clearly irrelevant here). In a formula,

$$(-\bar{f}_{\mathcal{B}})^*(z) = \min_{z',\rho} \left\{ \rho(-f_{\mathcal{B}})^*(z') + (1-\rho)c\tilde{x} \;:\; z = \rho z' + (1-\rho)(A\tilde{x} - b) \,,\; \rho \in [0,1] \right\} .$$

Therefore, the stabilized primal master problem (20) using model $\bar{f}_{\mathcal{B}}$ is

$$\min_{z,z',x',\theta'_{\mathcal{B}},\rho} \left\{ \begin{array}{l} \rho c x' + (1-\rho)c\tilde{x} - \bar{\pi} z + \mathcal{D}_t^*(-z) \\ z' = Ax' - b \,,\; x' = C_{\mathcal{B}}\theta'_{\mathcal{B}} \,,\; \Gamma_{\mathcal{B}}\theta'_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \\ z = \rho z' + (1-\rho)(A\tilde{x} - b) \,,\; \rho \in [0,1] \end{array} \right. .$$

We can apply simple algebra to eliminate z', but this still leaves ungainly bilinear terms $\rho x'$ in the problem (note that the terms $(1-\rho)c\tilde{x}$ and $(1-\rho)(A\tilde{x} - b)$ pose no problem instead, as $\tilde{x}$ is a constant). However, *provided that* $\{\, \theta_{\mathcal{B}} \;:\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}} \,\}$ *is compact*, these can be effectively eliminated by the variable changes $x = \rho x'$ and $\theta_{\mathcal{B}} = \rho\theta'_{\mathcal{B}}$, which leads to

$$\min_{z,x,\theta_{\mathcal{B}},\rho} \left\{ \begin{array}{l} cx + (1-\rho)c\tilde{x} - \bar{\pi} z + \mathcal{D}_t^*(-z) \\ x = C_{\mathcal{B}}\theta_{\mathcal{B}} \,,\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \rho\gamma_{\mathcal{B}} \\ z = Ax + (1-\rho)A\tilde{x} - b \,,\; \rho \in [0,1] \end{array} \right. . \tag{23}$$

The problems are clearly equivalent: from the compactness assumption, $\{\, \theta_{\mathcal{B}} \;:\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq 0 \,\} = \{\, 0 \,\}$, and therefore $\rho = 0 \Rightarrow \theta_{\mathcal{B}} = 0 \Rightarrow x = 0$ as expected. When $\rho > 0$ instead, one simply has $x' = x/\rho$ and $\theta'_{\mathcal{B}} = \theta_{\mathcal{B}}/\rho$, and the equivalence follows algebraically. The problem therefore needs very little and intuitive modification: the new variable ρ is a "knob" that allows to either pick the fixed solution $\tilde{x}$ ($\rho = 0$), or any solution in $X_{\mathcal{B}}$ ($\rho = 1$), or "anything in between".

For the purpose of finite termination, performing aggregations is dangerous: if $\tilde{x}$ is allowed to change at each iteration, the set of all possible models $\bar{f}_{\mathcal{B}}$ is *not* finite. Thus, one has (at least in theory) to resort to something like a *safe* β-strategy [16, Definition 4.9], which is simply one where the total number of aggregations is finite (however this be obtained).

3.4 Convergence Results

We now turn to the convergence results for the method. The standing assumptions here are i)—vi). A basic quantity in the analysis is

$$\Delta f = f_{\mathcal{B}}(\bar{\pi}) - f(\bar{\pi}) \geq 0,$$

i.e., the "approximation error" of the model $f_{\mathcal{B}}$ with respect to the true function f in the tentative point $\bar{\pi}$. It can be shown that if $\Delta f = 0$, then $\bar{\pi}$ and $(\tilde{x}, \tilde{z})$ are the optimal solutions to the "exact" stabilized problems

$$\begin{array}{c} \max_{\pi} \left\{ f(\pi) - \mathcal{D}_t(\pi - \bar{\pi}) \right\} \\ \min_{z,x} \left\{ cx - \bar{\pi}z + \mathcal{D}_t^*(-z) \ : \ z = Ax - b \, , \ x \in X \right\} \end{array} \tag{24}$$

(with $f_{\mathcal{B}} = f$, $X_{\mathcal{B}} = X$), respectively [16, Lemma 2.2]. This means that $\bar{\pi}$ is the best possible tentative point, in terms of improvement of the function value, that we can ever obtain unless we change either t or $\bar{\pi}$: in fact, it is immediate to realize that if $\Delta f = 0$, then the "sufficient ascent" condition (21) surely holds. The "inherent finiteness" of our Lagrangian function f allows us to prove that this has to happen, eventually.

Lemma 11 *Assume that vii) holds and a monotone β-strategy is employed, or an asymptotically blocked β-strategy is employed; then, after finitely many NSs, either a SS is performed, or the algorithm stops.*

Proof. By contradiction, assume that there exists a sequence of infinitely many consecutive NSs (i.e., the algorithm never stops and no SS is done). By v), during sequences of consecutive NSs, t is bounded away from 0; by [16, Lemma 5.6], the sequence $\{\tilde{z}_i\}$ is bounded. Clearly, a monotone β-strategy implies that the optimal value $v(20)$ of the stabilized primal master problem is nonincreasing; however, if vii) holds, then [16, Theorem 5.7] ensures that it is actually *strictly decreasing*. Since $\bar{\pi}$ and (at length) t are fixed, this means that no two iterations can have the same $\mathcal{B}$, but the total number of possible different bundles $\mathcal{B}$ is finite. Analogously, if an asymptotically blocked β-strategy is employed, during an infinite sequence of consecutive NSs, one has that (21) is (at length, see iii)) never satisfied, and this clearly implies $f(\bar{\pi}) < f_{\mathcal{B}}(\bar{\pi})$. But, at length, removals from $\mathcal{B}$ are inhibited, and by iv) at length, at least one item has to be added to $\mathcal{B}$ at every iteration; thus, $\mathcal{B}$ grows infinitely large, contradicting finiteness in Assumption 1. ■

Theorem 12 *Under the assumptions of Lemma 11, the sequence $\{f(\bar{\pi}_i)\}$ converges to the optimal value of (2) (possibly $+\infty$). If (2) is bounded above, then a subsequence of $\{\tilde{x}_i\}$ converges to an optimal solution of (1). If, furthermore, $m = 1$ and a safe β-strategy is used, then the S^2DW algorithm finitely terminates.*

Proof. The standing assumption of [16, §6], i.e., that either the algorithm finitely stops with an optimal solution or infinitely many SSs are performed, is guaranteed by Lemma 11. Then, the first point is [16, Theorem 6.4] (being a polyhedral function, f is $*$-compact). The second point comes from [16, Theorem 6.2], which proves that the sequence $\{\tilde{z}_i\}$ converges to 0; then, compactness of X implies that a convergent subsequence exists. The third point is [16, Theorem 6.6]. Note that the latter uses [16, Theorem 6.5], which apparently requires that $f_{\mathcal{B}}$ be the cutting-plane model. However, this is not actually the case: the required property is that $f_{\mathcal{B}}$ be a polyhedral function, and that there exists a "large enough" $\mathcal{B}$ such that $f_{\mathcal{B}} = f$, which clearly happens here. ■

Note that setting $m = 1$ as required by Theorem 12 to attain finite convergence may come at a cost in practice. In fact, this turns the S^{2}DW method into a "pure proximal point" approach, where the "abstract" stabilized problems (24) have to be solved to optimality before $\bar{\pi}$ can be updated (it is easy to check that with $m = 1$ a SS can only be performed when $\Delta f = 0$). This is most often not the best choice, computationally [5], and for good reasons. The issue is mostly theoretical; in practice, finite convergence is very likely even for $m < 1$. Furthermore, as observed in the comments to [16, Theorem 6.6], the requirement can be significantly weakened; what is really necessary is to ensure that only finitely many consecutive SSs are performed with $\Delta f > 0$. Thus, it is possible to use any $m < 1$, provided that some mechanism (such as setting $m = 1$ after a while) ensures that sooner or later a SS with $\Delta f = 0$ is performed; once this happens (being $m = 1$ or not), the mechanism can be reset and m can be set back to a value smaller than 1.

The extension to the case where X is *not* compact is straightforward, as in the non-stabilized case, provided of course that Assumption 8 holds. The theoretical analysis hardly changes; in particular, [16, Lemma 5.6] is unaffected, and [16, Theorem 5.7] is easily extended. One may loose the second point of Theorem 12 (asymptotic convergence of $\{\tilde{x}_i\}$), but this is of no concern here; *finite* convergence basically only relies on the fact that the total number of possible different bundles $\mathcal{B}$ is finite, so we basically only need to ensure that the same triplet $(\mathcal{B}, \bar{\pi}, t)$ is never repeated. The fact that $conv(X)$ ($\Rightarrow$ some variables θ in its definition) may not be bounded has no impact, provided that the "set of pieces out of which the formulation of $conv(X)$ is constructed" is finite.

Convergence can also be obtained under weaker conditions. For instance:

- Strong coercivity in ii) can be relaxed provided that, like in the non-stabilized case, $\mathcal{B}$ is "sufficiently large" to ensure that (20) has at least one feasible solution. This actually provides a possible use for the aggregated model $\bar{f}_{\mathcal{B}}$ of Section 3.3 in case one knows (from outside of the solution process) some $\tilde{x} \in conv(X)$ such that $A\tilde{x} = b$; this guarantees that $\rho = 1, \theta_{\mathcal{B}} = 0$, $x = z = 0$ is feasible to (23). Dually, (18) is always bounded above since $\bar{f}_{\mathcal{B}} \leq f_{\tilde{x}} = c\tilde{x}$.

- Using stabilizing that are not smooth at zero (the other part of ii)) is possible provided that $\mathcal{D}_t \to 0$ (pointwise) as the algorithm proceeds; basically, this turns S^{2}DW into a penalty approach to (1), as $\mathcal{D}_t^* \to I_{\{0\}}$. In this case, it is also possible to limit changes of the center $\bar{\pi}$, up to *never* changing it.

- Constraints on the handling of t can be significantly relaxed, up to allowing it to converge to zero, provided that $\mathcal{D}_t$ is "regular enough" as a function of t; for instance, $\mathcal{D}_t = (1/t)D$ for some D satisfying i) and ii).

- The descent test (21) can be weakened by using $v(20)$ in place of $f_{\mathcal{B}}(\bar{\pi}) - f(\bar{\pi})$, making it easier to declare a SS.

The reader interested in these details is referred to [16]. Yet, although not the most general, the above scheme is already flexible enough to accommodate many algorithmic details that have proven to be useful in some application:

- Condition iii) allows to refrain from moving $\bar{\pi}$ even if a "sizable" ascent could be obtained (only provided that this does not happen infinitely many times); this allows for alternative actions to be taken in response to a "good" step, e.g. increasing $t \Rightarrow$ "flattening" $\mathcal{D}_t$.

- Condition iv) allows to solve the separation subproblem at different points than $\bar{\pi}$, and/or not to add the resulting items to $\mathcal{B}$, at some iterations; this is useful for instance to accomodate a further search on the dual space "around" $\bar{\pi}$, such as a linear or curved search.

- Condition v) allows a great deal of flexibility in changing the parameter t which controls the stabilizing term; the only requirement is that changes must be inhibited at some point during very long sequences of NSs. Thus, as soon as a SS is performed, t can be set to *whatever* value without hindering the convergence of the algorithm. This allows for many different actual strategies for updating t (see for instance [15]), which are known to be instrumental for the practical efficiency of stabilized approaches.

3.5 Stabilizing Functions

The S^{2}DW algorithm is largely independent on the choice of the stabilizing term $\mathcal{D}_t$, provided that the above weak conditions are satisfied. Indeed, by looking at (20), one notices that the choice of $\mathcal{D}_t$ only impacts on the $\mathcal{D}_t^*(-z)$ term in the objective function, allowing for many different stabilizing functions to be tested at relatively low cost in the same environment.

A number of alternatives have been proposed in the literature for the stabilizing function $\mathcal{D}_t$ or, equivalently, for the primal penalty term $\mathcal{D}_t^*$. In all cases, $\mathcal{D}_t$ is separable on the dual, and therefore $\mathcal{D}_t^*$ is such on the primal, that is,

$$\mathcal{D}_t(\pi) = \textstyle\sum_{i=1}^m \Psi_t(\pi_i) \qquad\qquad \mathcal{D}_t^*(z) = \textstyle\sum_{i=1}^m \Psi_t^*(z_i)$$

where $\Psi_t : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is a family of functions. Notable examples are:

- *Trust Region/BoxStep*: This uses $\Psi_t = I_{[-t,t]}$, that is, it establishes a trust region of "radius t" around the current point. From the dual viewpoint, this corresponds to $\Psi_t^* = t|\cdot|$, i.e., to a linear stabilization.
- *Proximal stabilization*: This uses $\Psi_t = \frac{1}{2t}(\cdot)^2$, $\Psi_t^* = \frac{1}{2}t(\cdot)^2$. Therefore, both the primal and dual master problems are convex quadratic problems with separable quadratic objective function.
- *Linear-quadratic penalty function*: This uses

$$\Psi_{t,\varepsilon}^*(z) = t \begin{cases} z^2/\varepsilon & \text{if } -\varepsilon \le s \le \varepsilon \\ |z| & \text{otherwise} \end{cases} \qquad \Psi_{t,\varepsilon}(y) = \begin{cases} \frac{\varepsilon}{4t} y^2 & \text{if } -t \le y \le t \\ +\infty & \text{otherwise} \end{cases}$$

  i.e., a modification of the boxstep method where nonsmoothness at zero of $\mathcal{D}_t^*$ is avoided.

Actually, the treatment can be extended to the case when the stabilizing term depends on multiple parameters instead of just one. One example is the 5-*piecewise linear penalty function*

$$\Psi_t(\pi) = \begin{cases} +\infty & \text{if} \quad \pi \le -(\Gamma^- + \Delta^-) \\ -\varepsilon^-(\pi - \Delta^-) & \text{if} \quad -(\Gamma^- + \Delta^-) \le \pi \le -\Delta^- \\ 0 & \text{if} \quad -\Delta^- \le \pi \le \Delta^+ \\ +\varepsilon^+(\pi - \Delta^+) & \text{if} \quad \Delta^+ \le \pi \le (\Delta^+ + \Gamma^+) \\ +\infty & \text{if} \quad (\Delta^+ + \Gamma^+) \le \pi \end{cases}, \tag{25}$$

whose corresponding 4-piecewise primal penalty is

$$\Psi_t^*(z) = \begin{cases} -(\Gamma^- + \Delta^-)z - \Gamma^-\varepsilon^- & \text{if} \quad -(\zeta^- + \varepsilon^-) \le z \le -\varepsilon^- \\ -\Delta^- z & \text{if} \quad -\varepsilon^- \le z \le 0 \\ +\Delta^+ z & \text{if} \quad 0 \le z \le \varepsilon^+ \\ +(\Gamma^+ + \Delta^+)z + \Gamma^+\varepsilon^+ & \text{if} \quad \varepsilon^+ \le z \le (\zeta^+ + \varepsilon^+) \end{cases}. \tag{26}$$

In this case, $t = [\zeta^{\pm}, \varepsilon^{\pm}, \Gamma^{\pm}, \Delta^{\pm}]$ is a vector of parameters (see Figure 4). These and similar piecewise-linear stabilizing functions have been tested (for instance, in [5]), showing that appropriately setting their parameters may lead to faster convergence of stabilized algorithms (in that case, applied to column generation).

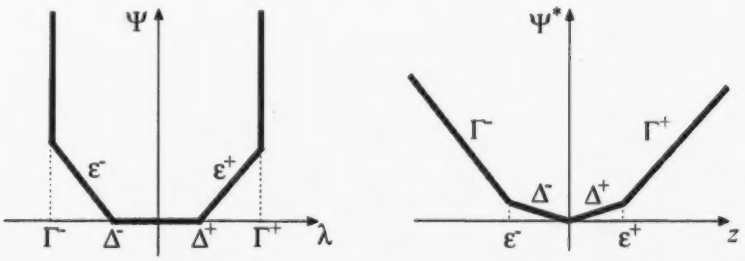

Figure 4: 5-piecewise (dual) and 4-piecewise (primal) stabilizing functions.

All these cases can be easily implemented within the same framework. Even the "complicated" 5-piecewise linear penalty case just corresponds to defining $z = z_2^- + z_1^- - z_1^+ - z_2^+$, with

$$\zeta^+ \geq z_2^+ \geq 0 \qquad \varepsilon^+ \geq z_1^+ \geq 0 \qquad \varepsilon^- \geq z_1^- \geq 0 \qquad \zeta^- \geq z_2^- \geq 0$$

in the primal master problem, with objective function

$$(\bar{y} - \Delta^- - \Gamma^-)z_2^- + (\bar{y} - \Delta^-)z_1^- - (\bar{y} + \Delta^+)z_1^+ - (\bar{y} + \Delta^+ + \Gamma^+)z_2^+ \ .$$

Hence, the primal master problem is still a linear program with the same number of constraints and $4m$ new variables. The other cases are even simpler.

4 Application to Multicommodity Capacitated Network Design

The Multicommodity Capacitated Network Design problem (MCND) we consider can be described as follows. Given a directed network $G = (N, A)$, where N is the set of nodes and A is the set of arcs, we must satisfy the communication demands between several origin-destination pairs, represented by the set of commodities K. Each commodity k is characterized by a positive communication demand d^k that must flow between the origin s_k and the destination t_k or, equivalently, by the *deficit vector* $b^k = [b_i^k]_{i \in N}$ with $b_i^k = -1$ if $i = s_k$, $b_i^k = 1$ if $i = t_k$, and $b_i^k = 0$ otherwise. While flowing along an arc (i, j), a communication consumes some of the *arc capacity*, which is originated by installing on the arc any number of *facilities*. Installing one facility on arc $(i, j) \in A$ provides a positive capacity u_{ij} at a (nonnegative) cost f_{ij}; a nonnegative routing cost c_{ij}^k also has to be paid for each unit of commodity k moving through (i, j). The problem consists in minimizing the sum of all costs, while satisfying demand requirements and capacity constraints.

We define nonnegative *flow variables* x_{ij}^k, which represent the fraction of the flow of commodity k on arc $(i, j) \in A$, i.e., $d^k x_{ij}^k$ is the flow of commodity k on arc (i, j). We also introduce general integer *design variables* y_{ij}, which define the number of facilities to install on arc (i, j). The problem

can then be formulated as follows:

$$\min \sum_{k\in K}\sum_{(i,j)\in A} d^k c_{ij}^k x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{27}$$

$$\sum_{(j,i)\in A} x_{ji}^k - \sum_{(i,j)\in A} x_{ij}^k = b_i^k \qquad i \in N\,,\ k \in K \tag{28}$$

$$\sum_{k\in K} d^k x_{ij}^k \le u_{ij} y_{ij} \qquad (i,j) \in A \tag{29}$$

$$0 \le x_{ij}^k \le 1 \qquad (i,j) \in A\,,\ k \in K \tag{30}$$

$$y_{ij} \ge 0 \text{ and integer} \qquad (i,j) \in A \tag{31}$$

We will denote this model as I. Its continuous relaxation $\bar{I}$ provides rather weak bounds, which translate into scarcely efficient enumerative approaches. It is therefore necessary to improve the bound if efficient approaches to the MCND are to be developed.

4.1 Dantzig-Wolfe Decomposition

Model I is amenable to solution through the DW approach in several different ways. We will consider the "complicating" constraints to be the flow conservation equations (28), and relax each of them with a Lagrangian multiplier π_i^k for each $k \in K$ and $i \in N$ (there are other possibilities, see for instance [8] for a related but different problem), resulting in a Lagrangian cost $\bar{c}_{ij}^k = d^k c_{ij}^k - \pi_i^k + \pi_j^k$ plus a fixed cost $\sum_{i\in N}\sum_{k\in K} \pi_i^k b_i^k$. This decomposes the problem into $|A|$ subproblems, one for each arc; therefore, to simplify the notation we will consider the arc index $(i,j) \in A$ fixed, and drop it. Each subproblem has the form

$$\min \textstyle\sum_{k\in K} \bar{c}^k x^k + fy \tag{32}$$

$$\textstyle\sum_{k\in K} d^k x^k \le uy \tag{33}$$

$$0 \le x^k \le 1 \qquad k \in K \tag{34}$$

$$y \ge 0 \text{ and integer} \tag{35}$$

which is easy to solve. Indeed, for any fixed value of $\bar{y}$ it reduces to the LP relaxation of a 0-1 knapsack problem, which can be solved efficiently (for instance, in $O(|K|\log|K|)$ through a sorting algorithm). Furthermore, if we relax the integrality constraint (35) and fix y to any given (continuous) value, the optimal value of the corresponding subproblem

$$z(y) = fy \;+\; \min_x \left\{ \textstyle\sum_{k\in K} \bar{c}^k x^k \;:\; (33) - (34) \right\}$$

defines a convex function of y (the partial minimization of a convex function is convex). Now, the global minimum $y^* \in \mathbb{R}$ of z can be easily found (see [1, 18] for details), and this allows to solve the problem. In fact, one just needs to consider $y^+ = \lceil y^* \rceil$ and $y^- = \lfloor y^* \rfloor$, compute $z(y^+)$ and $z(y^-)$, and pick whichever of the two is the best; it is easy to prove that this yields the optimal solution. Therefore, one can easily apply the standard DW approach, and in particular a ***disaggregated*** one; we will denote by DW the corresponding formulation (10) of the problem. Since (32)–(35), despite being easy to solve, does *not* have the integrality property, the bound computed by the DW approach is stronger than that of the standard continuous relaxation of I, i.e., $v(DW) \ge v(\bar{I})$ and a strict inequality usually holds (see Section 4.5).

4.2 Residual Capacity Inequalities

A different approach to improve the lower bound of a mixed-integer programming (MIP) formulation is to devise valid inequalities which cut out some of the fractional solutions of the ordinary continuous relaxation. For the MCND, the *residual capacity* inequalities [1, 26] have been developed which consider separately any single arc $(i,j) \in A$ (thus, once again we can drop the arc index). Then, for any subset $P \subseteq K$ of the commodities, one can define $d^P = \sum_{k \in P} d^k$, $a^P = d^P/u$, $q^P = \lceil a^P \rceil$, and $r^P = a^P - \lfloor a^P \rfloor$: the corresponding residual capacity inequalities can be written as

$$\sum_{k \in P} a^k (1 - x^k) \geq r^P (q^P - y) \ . \tag{36}$$

These inequalities are both valid and easy to separate for any given $(\bar{x}, \bar{y})$ where y is fractional. One simply defines $P = \{ \ k \in K \ : \ \bar{x}^k > \bar{y} - \lfloor \bar{y} \rfloor \ \}$ and checks if

$$\lfloor \bar{y} \rfloor < a^P < \lceil \bar{y} \rceil$$
$$\textstyle\sum_{k \in P} a^k (1 - \bar{x}^k - \lceil \bar{y} \rceil + \bar{y}) + \lfloor \bar{y} \rfloor (\lceil \bar{y} \rceil - \bar{y}) < 0$$

If these conditions are satisfied, then the residual capacity inequality corresponding to P is violated, otherwise there are no violated residual capacity inequalities. An interesting result is that residual capacity inequalities are "equivalent" to the DW approach; that is, denoting by $I+$ the formulation comprising all possible residual capacity inequalities and by $\bar{I}+$ its continuous relaxation, we have $v(\bar{I}+) = v(DW)$ (see [18] for details).

4.3 0-1 Reformulation

Yet another way to improve the quality of the lower bound is to devise an alternative 0-1 formulation of the MCND by means of a *multiple choice* model [9, 10]. Since $f_{ij} \geq 0$, we have $y_{ij} \leq \lceil \sum_{k \in K} d^k / u_{ij} \rceil = T_{ij}$ for each arc (i,j). Define $S_{ij} = \{1, \ldots, T_{ij}\}$, and introduce two sets of *auxiliary variables*

$$y_{ij}^s = \begin{cases} 1 & \text{if } y_{ij} = s \\ 0 & \text{otherwise} \end{cases} \qquad s \in S_{ij} \ , \ (i,j) \in A \tag{37}$$

$$x_{ij}^{ks} = \begin{cases} x_{ij}^k & \text{if } y_{ij} = s \\ 0 & \text{otherwise} \end{cases} \qquad s \in S_{ij} \ , \ (i,j) \in A, \ k \in K \ . \tag{38}$$

A 0-1 model of the problem can be obtained by first introducing the additional constraints

$$y_{ij} = \sum_{s \in S_{ij}} s y_{ij}^s \qquad (i,j) \in A \tag{39}$$

$$x_{ij}^k = \sum_{s \in S_{ij}} x_{ij}^{ks} \qquad (i,j) \in A \ , \ k \in K \tag{40}$$

$$(s-1) u_{ij} y_{ij}^s \leq x_{ij}^s \leq s u_{ij} y_{ij}^s \qquad (i,j) \in A \ , \ s \in S_{ij} \tag{41}$$

$$\sum_{s \in S_{ij}} y_{ij}^s \leq 1 \qquad (i,j) \in A$$

$$y_{ij}^s \in \{0,1\} \qquad (i,j) \in A \ , \ s \in S_{ij}$$

and then removing constraints (29) and (31) (which are implied by (39) and (41)) and projecting out variables y_{ij} and x_{ij}^k via (39) and (40), respectively. The corresponding model, which we will

denote as B, still provides the same (weak) bound as I. However, the *extended model* $B+$ obtained by adding the *extended linking constraints*

$$x_{ij}^{ks} \leq y_{ij}^{s} \qquad s \in S_{ij} \ , \ (i,j) \in A \ , \ k \in K$$

is stronger; indeed, for its continuous relaxation $\bar{B}+$ one has $v(\bar{B}+) = v(DW) = v(\bar{I}+)$ [18].

4.4 Summary of Algorithmic Approaches

The previous analysis shows that various algorithmic approaches exist for computing the same lower bound:

1. The DW approach applied to the original model I, either in the aggregated (4) or in the disaggregated (10) form; this is an LP with *exponentially many variables* and "few" constraints, solved by a variable generation technique.

2. The *Stabilized* DW (also known as bundle method) approach applied to the original model I, again in the two possible aggregated or disaggregated forms; this is a family of penalized problems, depending on the current point $\bar{\pi}$ and stabilizing parameter t. Each problem is usually either an LP or a QP with exponentially many variables and "few" constraints, which is approximately solved by a variable generation technique until $\bar{\pi}$ and/or t are changed.

3. The *Structured* DW approach applied to model $B+$, which is an LP with a *pseudo-polynomial* number of both variables and constraints, solved by *simultaneous* variable *and* constraint generation.

4. The *Stabilized Structured* DW approach applied to model $B+$; this is again a family of LPs or QPs depending on $\bar{\pi}$ and t, but each one now has a pseudo-polynomial number of variables and constraints, and it is approximately solved by variable and constraint generation until $\bar{\pi}$ and/or t are changed.

5. A completely different cutting-plane algorithm in the primal space, using residual arc capacity inequalities, applied to solve $\bar{I}+$: an LP with *exponentially many constraints* and "few" variables.

Clearly, apart from eventually providing the same bound and being all based on "very large" reformulations of the standard I model, these approaches bear little in common. The underlying reformulations have either very many columns and a few rows, or very many rows and relatively few columns, or an "intermediate" (albeit still very large) number of both rows and columns. The problems to be solved at each iteration may be either LPs or QPs, and be either very specially structured (so as to allow specialized approaches [14]) or rather unstructured. Implementing them may require little more than access to a general-purpose LP solver, such as in the case of 5 (where the dynamic generation of rows can be taken care of by the standard `callback` routines that all current LP solvers provide for the purpose), or access to somewhat more complex but still quite general codes for Lagrangian relaxation, such as in the case of 2, or, finally, development of entirely ad-hoc approaches such as for 3 and 4.

The algorithms may either be basically "fire and forget," or require nontrivial setting of algorithmic parameters. In particular, for stabilized approaches, the initial choice of $\bar{\pi}$ and the management of t can have a significant impact on performances. Regarding the first, in the absence of better options $\bar{\pi} = 0$ is the standard choice at the first iteration; however, for the MCND, better options

class	small	medium	large	huge
$\|N\|$	20	30	20	30
$\|K\|$	40	100	200	400

Table 1: Description of instances

exist. For instance, one may solve the $|K|$ separate shortest path problems corresponding to constraints (28) and (30), with $d^k c^k_{ij} + f_{ij}/u_{ij}$ as flow costs; this corresponds to solving the continuous relaxation of $\bar{I}$ (hence computing the weak bound), and produces *node potentials* $\bar{\pi}^k$ which can be used as starting point. Alternatively, or in addition, a few iterations of a subgradient-like approach can be performed to quickly get a better dual estimate.

Although all approaches compute the same relaxation bound, they should not be expected to be computationally equivalent. This indeed proves to be the case, as reported in the next section.

4.5 Computational Results

All experiments were performed on a computer with 16 Intel Xeon X7350 CPUs running at 2.93GHz and 64 Gb of RAM, running Linux Suse 11.1. All the LPs have been solved with CPLEX 11.1; in particular, the cutting-plane approach (5, above) has been implemented with the standard `cutcallback` of CPLEX. The stabilized DW approach to I (2, above) has been solved with a general-purpose bundle code developed by the first author and already used with success in several other applications [8, 19, 20]. The structure of the code allows to solve the Lagrangian dual with different approaches, such as the pure DW method (1, above) or "quick and dirty" subgradient-like methods [11], which have been shown to be competitive in some occasions [8, 20] for rapidly attaining rough estimates of the bound. The two other approaches (3 and 4, above) have required an ad-hoc implementation.

The experiments have been performed on 120 randomly generated problem instances, already used in [18]; the interested reader is referred there for more details. The instances are divided into four classes, whose characteristics are described in Table 1. For each class, 8 "basic" instances were generated and tested with 4 different values of the parameter

$$C = |A| \frac{\sum_{k \in K} d_k}{\sum_{(i,j) \in A} u_{ij}} \ ,$$

for a total of 32 instances; when $C = 1$ the average arc capacity equals the total demand and the network is lightly capacitated, while it becomes more tightly capacitated as C increases. All instances in a class share the same number of nodes and commodities, while the number of arcs may vary.

For non-stabilized approaches, the choice of the initial point is known [5] to have little impact on the performances. This is at large true also for the stabilized DW solved by the standard bundle code, which offers several sophisticated approaches for the critical on-line tuning of t [15, I.5]; whenever $\bar{\pi}$ is "far" form the optimum the strategies keep t "large," thereby allowing "long steps" and fast convergence, while when the optimum is approached t is reduced to enhance the locality properties. However, these strategies are not straightforward to adapt to the S^2DW setting, hence for that we kept t fixed for all the execution. As a consequence, the choice of the initial $\bar{\pi}$ had a significant impact on performances, and the (fixed) value of t had to be hand-tuned together with the choice of the initial point. Hence, apart from the shortest path warm-start (which can be used

by default since it is very inexpensive), we tested a two-level warm start where the shortest-path-produced $\bar{\pi}$ is further enhanced by a few iterations of a subgradient method. Some tuning of t and, for the latter, of the parameters of the subgradient, were necessary to select reasonable values; the parameters were chosen among a few choices, and kept fixed for all instances of the same class. In particular, the "optimal" t was shown to depend on the starting point, i.e., smaller (as expected) for the two-level warm start than for the standard shortest path warm-start, although only for two classes over four.

All the approaches only compute the same lower bound to the optimal value of the MCND, but of course this is only relevant as a step towards finding the optimal—or at least a provably good–solution to the problem. It is therefore interesting to gauge what is "the quality" of the partial formulation generated by each approach at termination. To do that, one way is to pick the final formulation produced by each approach and run CPLEX on it for one hour with the *polishing* option, in order to find a (hopefully, good) approximation of the best integer solution which is feasible for the partial formulation. This is easy with formulations $I+$ and $B+$, much less so (although in principle possible [27, 28]) with formulation DW. It is interesting to note that any restriction of $I+$ contains the optimal solution, while this is not true for $B+$.

We first ran an initial set of tests on a subset of the instances to get a first assessment of the effectiveness of each approach and to tune the algorithmic parameters where necessary; from these tests, we could conclude that:

- As expected, due to its instability, the standard (non stabilized) aggregated DW method could not reach the relative precision of `1e-6` in reasonable time, dramatically tailing off and effectively stopping to converge while still far from the expected value. Unlike in other applications [23], turning to the disaggregated model did not substantially change the situation.

- Not even the stabilized aggregated DW approach could reach convergence in reasonable time; while the method was indeed converging, the speed was exceedingly slow. This could be expected in view of the results on a similar problem [8], where a bundle method was found to be able to attain a not-too accurate estimate of the bound in reasonable time only by imposing strict bounds on the maximum size of $\mathcal{B}$. However, unlike in the non-stabilized case, the disaggregated variant improved things very substantially, resulting in a workable solution.

- The SDW method was always workable, although rather slow for very large instances. The S^2DW worked well, but there were significant differences in running time depending on the chosen stabilizing term; thus, we had to experiment both with the standard quadratic stabilization ($\Psi_t = 1/(2t)(\cdot)^2$) and with linear stabilization, namely, the three-pieces setting a trust region in the infinity norm of radius t around $\bar{\pi}$ ($\Psi_t = I_{[-t,t]}$).

We then ran a complete set of tests on the remaining approaches, i.e., the primal cutting-plane approach using residual capacity inequalities on the integer model I (denoted by "$I+$"), the (disaggregated) Stabilized Dantzig-Wolfe method on the integer model I (denoted by "StabDW"), the non-stabilized Structured Dantzig-Wolfe method on the binary model $B+$ (denoted by "StructDW"), and three versions of the Stabilized Structured Dantzig-Wolfe method on the binary model $B+$: with quadratic stabilization (denoted by "S^2DW_2"), with linear stabilization (denoted by "S^2DW_∞"), and with linear stabilization and two-level warm-start using subgradient (denoted by "S^2DW_∞-ws^2"). The results are shown in Tables 2 to 5. For each instance, we report the improvement (column "imp"), in percentage, between the "weak" lower bound $v(\bar{I})$ and the "strong" lower

bound $v(\bar{I}+) = v(DW) = v(\bar{B}+)$ computed by all methods. To compare the approaches, we report CPU times (column "cpu") and—with the exception of StabDW—the gap in percentage between the upper bound obtained as described above and the lower bound (column "gap"). For all approaches, we report the total number of iterations (column "it"); for S^2DW, we also report the number of Serious Steps (column "ss").

For small instances (Table 2), the six approaches are roughly equivalent. StabDW is typically slightly faster than all the others, but very close to $I+$. StructDW attains slightly worse performances in time, and very similar ones in gap for the cases (almost always $C = 16$) where it is non-null. Among the Stabilized SDW approaches, S^2DW_2 has the smallest number of iterations, slightly smaller than the non-stabilized approach, but it requires a longer running time due to the quadratic (vs. linear) master problem. The linear stabilization often *worsens* the iteration count significantly compared to both the quadratic and the non-stablized version, although the "sophisticated" warm-start definitely helps in clawing back some performances; however, while S^2DW_∞ is by far the worst performer in terms of running time, it is also invariably the best (sometimes by a wide margin) in terms of gap. Yet, for instances taking (much) less than 3 seconds to execute, the differences, although significant in relative terms, are almost negligible in absolute value.

The picture is significantly different already for medium instances (Table 3). First, due to a very large number of iterations, the StabDW approach is the fastest only for two (easy) instances, it is somewhat competitive in two other ones (except for $C = 16$), but loses badly on the most difficult ones. The Structured DW approach suffers a significant degradation of performances (up to five-fold) as C grows from 1 to 16, making it not competitive for the largest values of C, although it is one of the best approaches for small C. The quadratic S^2DW is no longer invariably better than the non-stabilized SDW in terms of iterations, except for $C = 16$; however, the cost of the quadratic master problem explodes, especially for some instances, making it very unattractive. We remark here that both the active-set ("simplex") and interior-point ("barrier") algorithms of CPLEX have been tested, with the former proving (as expected) more efficient due to its better reoptimization capabilities; yet, this was not sufficient to achieve good performances. The linear S^2DW is again worse than the quadratic one in iteration count, but much better in running time; furthermore, it is much more stable than the non-stabilzed SDW as C grows, so that while the latter is usually faster for $C \leq 4$, the reverse happens for $C \geq 8$. The effect of the two-stage warm-start is somewhat erratic in these instances, seldom being of any use in terms of CPU time; however, the gaps (where nonzero) are most often substantially reduced. The primal cutting-plane $I+$ is generally comparable to S^2DW_∞ in performances, often (but not always) being better; yet, in terms of gap S^2DW_∞ clearly dominates, especially in the most "difficult" instances where the final gaps are larger.

The trends seen for the medium instances are confirmed and amplified in the large ones (Table 4). The (disaggregated) Stabilized DW requires more than 100000 iterations on average, and therefore is typically very slow. The non-stabilized Structured DW suffers the same sharp degradation of performances as C increases, with almost an order of magnitude difference in one case. A completely opposite trend, barely discernible already in the "hard" medium instances, reveals itself for $I+$: the approach is much more efficient for large C than for small C, with the ratio between $C = 1$ and $C = 16$ reaching almost 25 in one case. Thus, while StructDW is much better than $I+$ for $C = 1$, typically by more than an order of magnitude, $I+$ wins in a few cases for $C = 16$, although at most by 50%. The trend is repeated almost identical for gaps: StructDW is better (often considerably so) for $C \leq 8$, while in a few cases $I+$ attains a better gap for $C = 16$. The quadratic S^2DW again outperforms all other approaches in terms of iteration count; however, the cost of the master problem explodes to intolerable levels. Furthermore, the gaps are not quite as good as those of

Problem			$I+$			StabDW		StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
$\|A\|$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss	cpu	gap	it	ss
230	1	14.87	0.14	0.00	6	0.10	185	0.22	0.00	9	0.24	0.00	7	6	0.82	0.00	52	47	0.43	0.00	18	14
	4	12.85	0.13	0.00	6	0.07	183	0.23	0.00	10	0.22	0.00	7	5	0.71	0.00	44	40	0.32	0.00	13	6
	8	10.62	0.14	0.00	6	0.10	222	0.30	0.00	12	0.26	0.00	8	5	0.82	0.00	49	42	0.37	0.00	11	9
	16	7.15	0.16	0.52	7	0.14	291	0.35	0.52	13	0.50	0.52	12	7	0.97	0.48	61	46	0.44	1.50	23	10
230	1	47.16	0.31	0.00	12	0.12	281	0.35	0.00	12	0.63	0.00	12	8	2.65	0.00	130	126	0.54	0.00	20	17
	4	37.54	0.26	0.00	10	0.16	309	0.44	0.00	15	0.52	0.00	10	7	2.26	0.00	110	108	0.44	0.00	13	12
	8	27.35	0.22	0.00	8	0.11	261	0.41	0.00	12	0.71	0.00	12	9	2.41	0.00	118	109	0.55	0.00	20	17
	16	15.69	0.23	1.45	9	0.48	667	0.58	1.86	18	0.87	2.17	14	8	2.26	0.46	111	96	0.71	0.42	30	20
230	1	7.70	0.10	0.00	4	0.06	166	0.29	0.00	17	0.20	0.00	6	5	0.43	0.00	29	25	0.25	0.00	7	3
	4	6.74	0.09	0.00	4	0.05	163	0.19	0.00	8	0.20	0.00	6	5	0.43	0.00	26	22	0.25	0.00	7	4
	8	5.68	0.10	0.00	4	0.07	181	0.26	0.00	13	0.24	0.00	7	6	0.38	0.00	23	20	0.25	0.00	6	3
	16	4.11	0.12	0.29	5	0.11	268	0.33	0.29	13	0.30	0.29	8	5	0.81	0.17	67	41	0.29	0.57	7	3
230	1	27.43	0.15	0.00	7	0.08	232	0.28	0.00	12	0.33	0.00	10	7	1.41	0.00	79	78	0.63	0.00	30	27
	4	23.06	0.18	0.00	8	0.12	225	0.26	0.00	10	0.29	0.00	7	6	1.41	0.00	80	76	0.48	0.00	25	13
	8	17.96	0.18	0.30	8	0.14	317	0.26	0.43	10	0.38	0.30	9	7	1.13	0.15	64	58	0.45	0.13	18	12
	16	11.50	0.16	1.20	7	0.28	473	0.43	1.20	14	0.59	1.20	12	7	1.70	0.45	95	75	0.95	1.62	62	30
289	1	20.91	0.18	0.00	6	0.09	193	0.27	0.00	11	0.22	0.00	6	4	0.80	0.00	47	45	0.67	0.00	27	25
	4	17.99	0.17	0.00	6	0.08	182	0.23	0.00	9	0.22	0.00	6	4	0.67	0.00	39	37	0.45	0.00	16	11
	8	14.47	0.18	0.09	6	0.10	213	0.29	0.09	10	0.29	0.09	7	6	0.70	0.09	42	38	0.33	0.07	8	4
	16	9.70	0.16	1.20	5	0.11	224	0.31	1.20	10	0.34	1.20	8	5	1.07	0.81	74	51	0.38	0.86	12	6
289	1	52.80	0.27	0.00	9	0.11	244	0.28	0.00	9	0.46	0.00	11	6	1.66	0.00	66	64	0.54	0.00	19	18
	4	42.57	0.27	0.00	9	0.11	246	0.35	0.00	12	0.52	0.00	11	8	0.89	0.00	44	35	0.88	0.00	37	36
	8	31.82	0.25	0.81	8	0.13	264	0.36	0.81	11	0.55	0.81	12	7	1.97	0.44	104	99	0.52	0.32	19	12
	16	17.68	0.27	2.50	8	0.13	260	0.48	2.70	14	0.67	3.05	13	8	1.87	1.38	93	84	0.47	1.34	16	8
289	1	11.26	0.15	0.00	5	0.08	187	0.22	0.00	10	0.21	0.00	6	4	0.49	0.00	29	26	0.29	0.00	7	2
	4	9.90	0.15	0.00	5	0.10	205	0.25	0.00	10	0.25	0.00	7	6	0.46	0.00	26	24	0.38	0.00	12	7
	8	8.07	0.14	0.00	4	0.08	208	0.29	0.00	10	0.21	0.00	7	4	0.40	0.00	25	20	0.31	0.00	10	3
	16	5.64	0.15	0.58	5	0.09	223	0.40	0.58	14	0.35	0.58	9	6	0.49	0.49	30	23	0.34	0.66	7	5
289	1	37.54	0.26	0.00	9	0.11	211	0.27	0.00	8	0.37	0.00	8	6	0.74	0.00	35	30	0.83	0.00	37	34
	4	31.36	0.25	0.00	8	0.10	223	0.25	0.00	9	0.37	0.00	9	6	0.63	0.00	31	26	0.63	0.00	26	24
	8	24.63	0.24	0.51	8	0.13	258	0.30	0.51	10	0.33	0.51	7	6	1.35	0.22	75	70	0.59	0.13	24	16
	16	14.93	0.19	1.95	6	0.14	268	0.49	2.02	15	0.40	2.04	9	5	1.34	0.67	73	64	0.43	0.88	13	7
avg.		19.65	0.19	0.36	7	0.12	251	0.32	0.38	12	0.38	0.40	9	6	1.13	0.18	62	55	0.48	0.27	19	13

Table 2: Results for small instances

Problem			$I+$			StabDW		StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞-ws^2			
$\|A\|$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss	cpu	gap	it	ss
517	1	76.58	9.33	0.14	21	31.14	3556	7.18	0.10	25	79.34	0.09	27	11	10.87	0.09	42	15	11.71	0.09	39	25
	4	68.29	8.55	0.09	20	31.90	3606	6.70	0.10	21	97.08	0.09	31	13	10.81	0.09	39	15	11.10	0.09	42	18
	8	58.46	11.69	0.09	24	30.83	3595	8.08	0.10	21	61.60	0.09	23	12	9.35	0.09	35	12	8.03	0.09	31	13
	16	43.01	11.26	0.43	20	46.07	3742	15.80	0.43	24	96.36	0.43	27	13	12.91	0.43	37	13	8.02	0.42	24	13
517	1	187.00	348.18	5.78	26	4323.41	88144	296.30	6.94	55	16380.00	6.57	51	15	223.22	2.97	66	58	357.38	1.52	91	84
	4	138.22	362.02	6.42	25	3581.13	79390	312.13	7.48	44	17091.70	5.87	47	12	298.34	2.72	70	54	269.85	1.48	69	60
	8	100.08	305.33	6.12	21	4054.19	88807	633.14	6.11	61	22176.20	7.16	37	14	279.88	2.70	64	34	276.91	1.44	65	47
	16	60.49	249.12	6.20	21	3015.71	71651	1138.46	6.45	87	27033.90	6.08	43	18	190.24	2.78	60	21	118.59	1.52	40	18
517	1	56.66	4.80	0.12	21	9.08	1519	3.73	0.03	21	11.68	0.03	28	12	5.30	0.03	35	12	8.34	0.03	52	22
	4	51.68	4.86	0.03	21	8.57	1509	3.97	0.03	21	9.96	0.03	22	12	4.89	0.03	33	12	3.97	0.03	22	10
	8	45.42	5.16	0.03	22	8.43	1488	5.06	0.03	25	9.92	0.03	21	11	4.26	0.03	26	10	5.43	0.03	31	14
	16	35.19	4.74	0.69	18	33.65	3777	8.16	0.69	29	15.04	0.73	27	13	6.68	0.52	36	13	3.83	0.43	19	8
517	1	155.19	140.92	3.95	23	2899.18	69500	188.22	4.70	60	5802.88	4.01	42	13	204.94	2.56	71	57	222.32	1.43	85	71
	4	122.84	194.00	3.87	26	2799.31	65229	147.30	4.15	39	6453.45	4.32	39	15	215.22	2.43	79	40	91.40	1.39	41	36
	8	93.00	151.01	3.96	20	2823.68	66025	354.67	4.31	67	5752.64	4.40	31	12	166.92	2.38	62	25	124.17	1.42	50	21
	16	59.68	115.99	4.72	18	2171.57	56184	551.12	4.94	70	10154.30	5.07	40	14	162.76	2.76	61	20	113.23	1.53	50	19
669	1	74.12	6.27	0.00	14	2.62	655	4.94	0.00	19	40.72	0.00	26	13	8.67	0.00	40	16	13.78	0.00	54	33
	4	66.28	6.98	0.00	15	2.74	657	6.75	0.00	30	50.74	0.00	30	16	7.09	0.00	34	13	8.40	0.00	31	23
	8	57.33	6.90	0.00	15	3.06	723	6.98	0.00	23	37.37	0.00	24	13	7.72	0.00	35	13	7.51	0.00	42	12
	16	43.23	6.05	0.02	13	3.16	803	10.63	0.02	27	33.36	0.02	26	13	9.08	0.02	42	10	8.26	0.02	41	13
669	1	114.50	80.33	0.50	26	330.03	11273	36.49	0.46	32	2405.96	0.46	47	15	84.37	0.41	76	48	77.74	0.33	72	66
	4	97.32	78.24	0.46	22	326.88	10951	66.31	0.46	50	1964.43	0.46	45	14	66.86	0.41	74	24	81.00	0.33	73	56
	8	79.62	68.01	0.46	19	322.55	11173	55.29	0.46	33	1974.25	0.46	44	15	49.63	0.41	57	18	39.59	0.33	49	20
	16	56.19	58.16	0.74	19	274.54	9979	164.48	0.81	65	1408.34	0.80	38	17	47.33	0.61	52	16	44.39	0.40	52	22
669	1	55.23	4.59	0.00	16	0.58	246	3.13	0.00	18	8.47	0.00	20	11	5.25	0.00	34	13	6.38	0.00	30	20
	4	50.50	4.34	0.00	16	0.61	246	2.97	0.00	16	6.93	0.00	14	10	5.09	0.00	30	13	5.49	0.00	33	13
	8	44.58	3.66	0.00	13	0.61	265	4.08	0.00	22	7.58	0.00	17	11	5.03	0.00	31	12	5.50	0.00	36	12
	16	34.97	4.41	0.16	15	3.13	745	6.30	0.16	29	10.86	0.16	21	12	5.24	0.16	31	12	4.16	0.17	27	8
669	1	88.97	21.42	0.51	24	27.17	2714	13.97	0.25	33	416.23	0.25	45	19	29.65	0.25	84	28	31.46	0.16	65	58
	4	78.13	19.65	0.25	22	28.88	2768	13.60	0.25	31	293.43	0.25	40	13	20.22	0.25	53	20	22.85	0.16	55	38
	8	66.21	22.23	0.25	21	30.89	2799	18.33	0.25	30	234.38	0.25	39	12	15.64	0.25	40	15	12.17	0.16	37	15
	16	48.93	18.06	0.32	17	244.94	9618	33.94	0.32	33	222.54	0.32	37	16	14.25	0.32	35	12	13.62	0.25	38	15
avg.		75.25	73.01	1.45	20	858.45	21042	129.01	1.56	36	3760.68	1.51	33	13	68.37	0.80	49	22	63.02	0.48	46	28

Table 3: Results for medium instances

Problem			$I+$			StabDW		StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
$\|A\|$	C	imp	cpu	gap	it	cpu	it	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss	cpu	gap	it	ss
229	1	162.59	6842	14.94	70	8525	117458	279	7.29	43	5446	8.55	24	11	449	3.10	71	64	359	1.41	86	66
	4	117.19	6661	13.55	71	8796	115025	441	9.02	58	10997	8.02	36	17	332	3.09	55	42	466	1.38	73	62
	8	83.49	4850	9.92	63	9098	119791	978	8.51	57	19742	7.29	26	12	424	3.02	55	36	381	1.36	57	43
	16	48.68	1651	8.29	50	5785	83284	1813	7.15	75	38765	7.48	37	15	372	2.37	60	22	284	1.10	52	28
229	1	205.67	49081	28.16	109	11748	154821	525	10.50	44	17660	12.11	32	17	860	4.16	76	73	907	1.32	129	119
	4	131.24	30899	25.40	91	9132	131674	807	13.58	45	27326	10.20	29	15	1091	2.79	89	87	1460	1.23	126	116
	8	84.61	16502	21.80	87	12682	162766	1593	10.17	44	83226	10.12	40	17	1027	3.03	78	61	1237	1.20	99	77
	16	42.78	2090	5.59	54	6541	97952	2630	9.20	73	108453	9.21	54	16	399	2.12	65	31	804	1.02	114	73
229	1	147.99	3255	11.58	58	7100	106198	231	7.94	41	4030	7.57	24	12	309	2.78	62	50	242	1.25	67	56
	4	110.93	3198	12.38	57	8114	118176	219	7.82	34	6684	6.88	31	16	300	3.00	57	38	297	1.21	65	49
	8	80.92	2917	7.32	56	8844	110752	510	7.77	41	13273	6.70	37	13	242	2.69	41	30	281	1.28	54	39
	16	49.19	1300	5.87	49	7748	106260	1388	7.09	62	33357	7.81	39	19	249	2.15	42	17	251	1.06	53	28
229	1	185.17	18326	20.53	86	9261	132963	380	7.44	39	10173	****	29	14	557	2.61	80	71	592	1.30	101	95
	4	125.39	15537	18.81	80	11791	147879	612	9.36	49	12638	10.33	25	15	755	2.87	80	68	930	1.22	98	93
	8	85.31	9500	13.08	74	10702	146727	1647	8.87	68	32405	10.61	30	14	468	2.75	50	43	761	1.33	83	66
	16	46.09	1900	7.19	52	7268	107197	3167	7.99	108	69562	8.32	47	17	476	2.22	67	30	357	1.10	53	39
287	1	152.25	3186	12.01	52	8938	122231	269	8.77	42	6791	****	30	13	587	3.51	79	66	378	1.55	74	69
	4	118.18	3119	11.77	56	8622	116606	275	8.56	38	6187	8.05	28	12	272	3.59	47	35	372	1.63	66	62
	8	88.39	2887	11.21	51	6707	99283	668	8.14	61	14935	8.69	33	14	463	3.61	57	40	328	1.52	49	40
	16	55.39	1233	7.11	35	7327	99542	1795	7.80	92	46638	6.38	41	16	345	3.01	56	19	230	1.43	40	21
287	1	198.87	14559	27.86	66	8815	120614	598	12.54	53	20949	16.31	39	15	1019	3.92	98	93	1327	1.65	149	143
	4	136.97	11934	22.52	62	8426	112308	603	15.07	37	18258	13.78	27	15	1001	3.72	90	79	891	1.60	98	94
	8	92.94	9656	15.28	64	10098	130536	1221	10.38	41	51703	11.81	29	14	909	3.68	73	50	1040	1.63	102	96
	16	53.45	3579	11.60	54	6801	98972	3515	9.06	99	132097	10.11	54	17	513	2.93	59	25	555	1.26	62	45
287	1	144.47	2082	11.33	47	7398	104285	241	7.30	45	5892	7.09	36	13	348	3.46	60	48	404	1.63	81	73
	4	114.14	2141	8.34	49	7234	107048	207	9.13	34	7166	7.35	35	14	240	3.55	44	36	328	1.65	66	59
	8	86.69	1723	10.36	42	6125	90230	445	9.20	50	11131	****	38	16	340	3.22	52	31	277	1.51	47	39
	16	55.10	1049	7.17	34	6154	91064	1180	6.96	73	29143	6.88	40	14	297	3.20	50	17	165	1.48	35	18
287	1	190.82	13162	18.61	74	11493	152106	577	13.15	54	15691	12.53	39	13	1028	3.76	97	91	749	1.67	107	96
	4	134.41	12015	25.90	72	12067	151753	829	13.04	65	21437	****	40	13	686	3.88	71	60	919	1.74	101	96
	8	92.71	8032	15.56	61	9911	130539	1478	12.58	60	88613	14.62	45	17	759	3.64	66	50	680	1.59	66	55
	16	54.21	2533	8.82	44	6945	103842	1936	8.59	59	133781	8.12	58	17	436	2.85	51	27	455	1.44	52	40
avg.		108.63	8356	14.06	62	8642	118434	1033	9.37	56	34505	20.72	36	15	549	3.13	65	48	585	1.40	78	66

Table 4: Results for large instances

Problem			StabDW		StructDW			S^2DW_∞				S^2DW_∞-ws^2			
$\|A\|$	C	imp	cpu	it	cpu	gap	it	cpu	gap	it	ss	cpu	gap	it	ss
519	1	100.83	87695	248746	9839	9.96	157	2473	2.23	76	55	1857	2.31	53	38
	4	92.54	88031	247864	9087	11.25	140	2140	2.33	68	54	2487	2.36	66	44
	8	82.16	88918	258266	11613	8.47	143	2338	2.45	66	45	1813	2.30	52	30
	16	65.53	85384	238945	38617	10.26	242	3403	2.66	77	39	2570	2.26	58	23
519	1	140.14	95890	267645	21405	18.41	115	8741	3.14	107	94	7576	3.39	80	67
	4	121.46	103067	266754	25651	18.35	121	11101	3.17	118	94	7722	3.07	75	59
	8	101.19	95873	273728	47618	20.57	160	10020	3.00	103	74	8904	3.03	75	49
	16	73.67	92737	240947	60050	20.04	89	8245	3.00	72	52	8004	3.29	63	24
519	1	88.94	61543	187836	5153	5.75	128	1482	2.19	63	52	1139	1.91	49	32
	4	82.77	60911	188733	8207	6.25	205	1320	1.87	60	41	1341	1.77	53	32
	8	74.84	62377	187785	8490	4.87	163	1649	1.90	71	40	1262	1.84	47	30
	16	61.42	75193	220169	17422	6.35	202	1636	1.99	59	33	1522	2.11	55	19
519	1	125.07	93065	258054	22246	14.90	165	4811	3.31	87	76	4668	3.06	66	55
	4	111.02	90573	250854	17976	18.22	131	4324	2.57	77	64	4373	3.19	66	45
	8	94.82	93418	256884	30460	18.18	159	5224	3.14	85	60	4209	2.86	57	36
	16	71.31	93567	265663	74447	16.50	176	5532	3.14	67	46	5191	3.02	64	23
668	1	126.02	98789	246702	23771	11.89	149	9215	2.96	97	78	6815	3.01	69	56
	4	115.29	99014	247620	28567	10.97	176	6766	2.99	79	63	6506	3.07	69	45
	8	102.03	104481	258636	27871	12.07	130	7560	2.67	87	56	5765	2.78	61	37
	16	80.96	103011	278905	58363	13.95	156	8626	3.14	83	45	3764	2.95	41	18
668	1	111.16	92855	243448	13119	8.69	120	4316	2.53	77	63	4301	2.52	71	44
	4	103.21	94363	243804	12586	8.27	114	4117	2.36	79	51	2695	2.55	46	34
	8	93.15	88587	250759	20560	9.20	164	6314	2.31	102	64	3203	2.54	59	21
	16	76.10	101200	270216	38503	8.75	184	6360	2.30	95	44	3441	2.50	55	19
avg.		95.65	89606	246055	26317	12.17	154	5321	2.64	81	58	4214	2.65	60	37

Table 5: Results for huge instances

the linearly stabilized versions; in particular, in four cases (marked with "****" in the Table), no feasible solution at all was found. Here, the linear S^2DW really outperforms the competition: it is outperformed only by the non-stabilized SDW for small C, but it is much less affected by the growth of C (actually, most often than not it behaves better for large C than for small ones), being a factor of two faster on average. Furthermore, the gap is very substantially smaller than both StructDW and $I+$, irrespective of C. Again, the two-level warm-start has an erratic effect on running times, resulting in a very close average, but halves the gap when compared to the already surprisingly good result of the standard warm-start; this results in an average gap of 1.4%, which is a full order of magnitude less than the 14% gap obtained by $I+$.

For huge instances (Table 5), we did not compute results for $I+$ and S^2DW_2, as it is clear from the previous data that they have no hope to be competitive (for $I+$, this had already been shown in [18] when comparing it to StructDW alone). The Stabilized DW requires around 250000 iterations to converge, ending up the slowest in all cases. The non-stabilized Structured DW suffers from the same dramatic performance decline as C grows, making even the Stabilized DW on the original cutting-plane model competitive for $C = 16$. However, S^2DW is much more efficient in time, up to over one order of magnitude, and still delivers much smaller gaps. The two-level warm-start does not have the same uniform effect on gaps than for the medium and large instances, ending up

with very close results; however, in this case, the effect on running times is more noticeable, with a reduction between 20% and 50% being the most common outcome.

These results confirm that using the "more sophisticated" model of the Structured DW is typically very beneficial. The standard cutting-plane model, even the disaggregated one, may require a huge number of iterations to converge, thereby finally being less efficient despite the much lower cost of the master problem. This is analogous to what has been reported several times [2, 7, 23] to happen with the aggregated vs. disaggregated cutting-plane model: the much faster rate of convergence due to a more complex model can more than make up for the higher master problem cost. For "easy" (loosely capacitated) medium-to-large-scale instances, not stabilizing may be the best choice; this has been reported before for column generation [5]. However, for more tightly capacitated instances, especially as the size grows, stabilization becomes instrumental, provided that care is exercised in choosing the right stabilizing term. The effect on the gap is also noteworthy, since stabilization ends up not only being faster, but also collecting a smaller subset of the model, within which it is apparently much easier to find integer solutions of good quality. In other words, stabilization seems capable of helping the Structured Dantzig-Wolfe approach to select "just the right parts" of the huge model, not only in terms of the solution of the continuous relaxation, but also in terms of the integer solution. Exactly why this capability does not appear to be shared by the quadratically-stabilized version, that is otherwise clearly superior in terms of convergence speed, is not clear to us at this point in time, although one may speculate that the linear stabilization "changes less" the original linear objective function of the master problem. Thus, stabilization not only improves bound computation times, but it also appears—at least in this application—useful for constructing actual solutions of the original combinatorial program. Since the non-stabilized Structured Dantzig-Wolfe approach is a special case of the Stabilized one with an "infinitely weak" stabilization term, it is arguably convenient to approach any potential new application with a "stabilize first, ask questions later" strategy.

5 Conclusions and Future Research

We have proposed, analyzed and implemented a new extension of the Dantzig-Wolfe decomposition method. Our Structured Dantzig-Wolfe decomposition method improves on the original DW algorithm by allowing to exploit available information about the structure of the "easy" polyhedron under the form of a *reformulation* of the pricing problem amenable to a variable generation procedure. If the required assumptions are satisfied, the SDW approach has similar convergence properties than the original DW approach; furthermore, the subproblem that has to be solved in the two algorithms is the same—only the master problem changes—and therefore only limited modifications to existing DW approaches are required to implement SDW methods. Furthermore, the SDW method can be *stabilized* exactly like the original DW approach, leading to the Stabilized Structured Dantzig-Wolfe method (S^2DW); the convergence theory of [16] can be extended to the new approach, providing a reasonably complete picture about what stabilizing terms can be used, how the proximal parameter t and the "bundle" $\mathcal{B}$ can be handled, and so on. We have tested the S^2DW approach on an important combinatorial problem, the Multicommodity Capacitated Network Design problem, obtaining quite encouraging results against a large set of possible alternative approaches providing the same (strong) lower bound.

As far as future developments go, it will be interesting to collect more applications where the S^2DW approach can be applied. The S^2DW approach does not have the same wide applicability as the original DW approach, since it requires the existence of a reformulation of the "easy" problem with specific properties. However, we are confident that many more applications of the approach

can be constructed, with some (see for instance [6]) being already clearly identified as potential candidates; it will be interesting to verify in how many cases the new approach is competitive with existing solution methods. The implementation we tested is also rather naive in terms of the handling of t (fixed) and of $\mathcal{B}$ (no removals, no aggregation); finding appropriate rules for these important aspects has the potential to further substantially improve the computational behavior of the approach.

Acknowledgments

We are grateful to Serge Bisaillon for his help with implementing and testing the algorithms. We also gratefully acknowledge financial support for this project provided by NSERC (Canada) and by MIUR (Italy) under the PRIN projects "Optimization, simulation and complexity in telecommunication network design and management" and "Models and algorithms for robust network optimization".

References

[1] A. Atamtürk and D. Rajan. On Splittable and Unsplittable Flow Capacitated Network Design Arc-Set Polyhedra. *Mathematical Programming*, 92:315–333, 2002.

[2] L. Bacaud, C. Lemaréchal, A. Renaud, and C. Sagastizábal. Bundle Methods in Stochastic Optimal Power Management: A Disaggregated Approach Using Preconditioners. *Computational Optimization and Applications*, 20:227–244, 2001.

[3] L. Bahiense, N. Maculan, and C. Sagastizábal. The volume algorithm revisited: relation with bundle methods. *Mathematical Programming*, 94(1):41–70, 2002.

[4] G. Belov, G. Scheithauer, C. Alves, and J.M. Valério de Carvalho. Gomory Cuts from a Position-Indexed Formulation of 1D Stock Cutting. In A. Bortfeldt, J. Homberger, H. Kopfer, G. Pankratz, and R. Strangmeier, editors, *Intelligent Decision Support: Current Challenges and Approaches*, pages 3–14. Gabler, 2008.

[5] H. Ben Amor, J. Desrosiers, and A. Frangioni. On the Choice of Explicit Stabilizing Terms in Column Generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009.

[6] H. Ben Amor and J.M. Valério de Carvalho. Cutting Stock Problems. In J. Desrosiers G. Desaulniers and M.M. Solomon, editors, *Column Generation*, pages 131–161. Springer, 2005.

[7] A. Borghetti, A. Frangioni, F. Lacalandra, and C.A. Nucci. Lagrangian Heuristics Based on Disaggregated Bundle Methods for Hydrothermal Unit Commitment. *IEEE Transactions on Power Systems*, 18(1):313–323, 2003.

[8] T.G. Crainic, A. Frangioni, and B. Gendron. Bundle-based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems. *Discrete Applied Mathematics*, 112:73–99, 2001.

[9] K.L. Croxton, B. Gendron, and T.L. Magnanti. A Comparison of Mixed-Integer Programming Models for Non-Convex Piecewise Linear Cost Minimization Problems. *Management Science*, 49:1268–1273, 2003.

[10] K.L. Croxton, B. Gendron, and T.L. Magnanti. Variable Disaggregation in Network Flow Problems with Piecewise Linear Costs. *Operations Research*, 55:146–157, 2007.

[11] G. d'Antonio and A. Frangioni. Convergence Analysis of Deflected Conditional Approximate Subgradient Methods. *SIAM Journal on Optimization*, 20(1):357–386, 2009.

[12] G.B. Dantzig and P. Wolfe. The Decomposition Principle for Linear Programs. *Operations Research*, 8:101–111, 1960.

[13] L.R. Ford and D.R. Fulkerson. A Suggested Computation for Maximal Multicommodity Network Flows. *Management Science*, 5:79–101, 1958.

[14] A. Frangioni. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computers & Operations Research*, 21:1099–1118, 1996.

[15] A. Frangioni. *Dual-Ascent methods and Multicommodity flow problems*. PhD thesis, TD 5/97, Dipartimento di Informatica, Università di Pisa, Pisa, Italy, 1997.

[16] A. Frangioni. Generalized Bundle Methods. *SIAM Journal on Optimization*, 13(1):117–156, 2002.

[17] A. Frangioni. About Lagrangian Methods in Integer Optimization. *Annals of Operations Research*, 139:163–193, 2005.

[18] A. Frangioni and B. Gendron. 0-1 Reformulations of the Multicommodity Capacitated Network Design Problem. *Discrete Applied Mathematics*, 157(6):1229–1241, 2009.

[19] A. Frangioni, C. Gentile, and F. Lacalandra. Solving Unit Commitment Problems with General Ramp Contraints. *International Journal of Electrical Power and Energy Systems*, 30:316 – 326, 2008.

[20] A. Frangioni, A. Lodi, and G. Rinaldi. New approaches for optimizing over the semimetric polytope. *Mathematical Programming*, 104(2-3):375–388, 2005.

[21] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms* I—*Fundamentals*, volume 306 of *Grundlehren Math. Wiss.* Springer-Verlag, New York, 1993.

[22] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms* II—*Advanced Theory and Bundle Methods*, volume 306 of *Grundlehren Math. Wiss.* Springer-Verlag, New York, 1993.

[23] K.L. Jones, I.J. Lustig, J.M. Farwolden, and W.B. Powell. Multicommodity Network Flows: The Impact of Formulation on Decomposition. *Mathematical Programming*, 62:95–117, 1993.

[24] J.E. Kelley. The Cutting-Plane Method for Solving Convex Programs. *Journal of the SIAM*, 8:703–712, 1960.

[25] C. Lemaréchal. Lagrangian Relaxation. In Jünger, M. and Naddef, D., editors, *Computational Combinatorial Optimization*, pages 115–160. Springer-Verlag, Heidelberg, 2001.

[26] T.L. Magnanti, P. Mirchandani, and R. Vachani. The Convex Hull of Two Core Capacitated Network Design Problems. *Mathematical Programming*, 60:233–250, 1993.

[27] F. Vanderbeck. On Dantzig-Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-Price Algorithms. *Operations Research*, 48(1):111–128, 2000.

[28] D. Villeneuve, J. Desrosiers, M.E. Lübbecke, and F. Soumis. On Compact Formulations for Integer Programs Solved by Column Generation. *Annals of Operations Research*, 139:375–388, 2005.